Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Luka Hlastec

Razvoj sistema za krmiljenje garažnih vrat s pomočjo mobilne aplikacije

diplomsko delo

visokošolski strokovni študijski program prve stopnje računalništvo in informatika

Mentor: prof. dr. Nikolaj Zimic

Ljubljana 2014

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

S pocenitvijo elektronskih sklopov je postala hišna avtomatizacija cenovno široko dostopna. Tehnologija nam omogoča z majhnimi stroški postaviti spletni strežnik, ki upravlja s hišnimi sistemi.

V diplomski nalogi izdelajte sistem za krmiljene garažnih vrat. Pri tem uporabite običajni mehanizem za odpiranje in zapiranje vrat, ki omogoča upravljanje preko tipke za odpiranje in zapiranje. Sistem naj bo priključen v internet, kar bo uporabnikom omogočilo upravljanje preko mobilne aplikacije.

V prvem delu naloge povežite mikroračunalnik Raspberry PI neposredno z mehanizmom za opiranje in zapiranje vrat. Pri tem namestite manjkajoče senzorje za odprtost in zaprtost vrat. V drugi fazi razvoja strojne opreme zamenjajte žično povezavo mehanizma vrat z brezžično.

V drugem delu diplomske naloge izdelajte ustrezno programsko opremo, ki bo omogočala enostavno upravljanje vrat preko interneta. Pri tem poskrbite za ustrezne varnostne mehanizme, ki bodo uporabnikom zagotavljali ustrezno stopnjo varnosti.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Luka Hlastec, z vpisno številko **63990204**, sem avtor diplomskega dela z naslovom:

Razvoj sistema za krmiljenje garažnih vrat s pomočjo mobilne aplikacije

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom prof. dr. Nikolaja Zimica,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 15. maja 2014 Podpis avtorja:

Za pomoč in nasvete pri izdelavi diplomske naloge se iskreno zahvaljujem mentorju prof. dr. Nikolaju Zimicu.
Posebna zahvala za razumevanje in podporo v času študija je namenjena tudi staršem in punci.

Kazalo

Povzetek

V diplomski nalogi je obravnavan razvoj sistema Vratar, katerega funkcionalnost omogoča krmiljenje pogona garažnih vrat s pomočjo mobilne aplikacije. Za ta namen je bil postavljen strežnik, ki temelji na računalniku Raspberry Pi. V okviru strežniškega dela je bila razvita aplikacija za upravljanje sistema vrat. Za komunikacijo s sistemom se uporablja splošno namenski vmesnik. Poleg osnovne različice sistema je predstavljena še nadgradnja, pri kateri je komunikacija s sistemom vrat realizirana z uporabo radijske povezave. Opisan je tudi razvoj odjemalca za mobilno platformo iOS. Predstavljene so posamezne funkcionalnosti odjemalca, kot so potisna obvestila, nadzor vrat preko mrežne kamere in med drugim tudi odpiranje vrat z uporabo lokacijskih storitev. Diplomska naloga se zaključi s predstavitvijo ugotovitev in možnosti nadaljnjega razvoja sistema Vratar.

Ključne besede: iOS, Vratar, mVratar, Raspberry Pi, Arduino, garažna vrata.

Abstract

This thesis covers the development of the system Vratar which enables controlling the drive of garage doors with the use of a mobile application. A server, which is based on the computer Raspberry Pi, was set up for this purpose. In the scope of the server part an application was developed for controlling the door system. A general purpose interface was used for communication. Apart from the basic version of the system, this thesis also presents an upgrade where the communication with the door system is realized with the use of a radio connection. The development of a client for the mobile platform iOS is also described. Individual functions of the client such as push notifications, door surveillance via web camera and also opening of doors via location services are presented. The thesis finishes with a presentation of findings and possibilities for further development of the door system.

Keywords: iOS, Vratar, mVratar, Raspberry Pi, Arduino, garage door.

Poglavje 1

Uvod

Področje hišne avtomatizacije je v današnjem času zelo priljubljeno. V veliki meri je k temu pripomogla pocenitev elektronskih komponent, ki postajajo vse bolj dostopne tudi domačemu uporabniku. V diplomski nalogi smo izdelali sistem za krmiljenje avtomatiziranih garažnih vrat s pomočjo mobilne aplikacije, ki smo jo poimenovali mVratar.

Diplomska naloga se prične z opisom celotnega sistema Vratar. V nadaljevanju sledi opis strojne opreme, katero smo potrebovali za realizacijo sistema Vratar. Ena izmed ključnih komponent je računalnik Raspberry Pi. Gre za cenovno ugoden mini računalnik v velikosti kreditne kartice, ki temelji na procesorju podjetja Broadcom. Odlikujejo ga številni vmesniki, med njimi tudi splošno-namenski vmesnik GPIO (angl. General Purpose Input/Output), ki se uporablja za komunikacijo s strojno opremo. V diplomski nalogi smo računalnik Raspberry Pi uporabili kot strežnik. Nanj smo namestili aplikacijo, ki preko vmesnika GPIO komunicira s senzorji in mehanizmom za krmiljenje vrat. Te smo sprva povezali preko ustreznega vmesnika na vmesnik GPIO. Ker se to v nekaterih primerih ni izkazalo za najbolj uporabno, vrata so lahko namreč od strežnika oddaljena več sto metrov, smo razvili še brezžično različico sistema. V tej različici poteka komunikacija med strežnikom in vozlišči preko radijske povezave. Tu je bilo potrebno zagotoviti, da je vsak ukaz, ki se pošilja preko radijskega signala, edinstven. S tem preprečimo možnost, da bi nekdo prestregel radijski signal in si s ponovnim predvajanjem signala odprl vrata. Problem smo rešili z implementacijo gesel, katerih veljavnost je časovno omejena. Ta gesla se dodajo ukazom, ki se pošiljajo preko radijskega

signala. Vozlišče pri sprejemu ukaza za krmiljenje vrat najprej preveri veljavnost gesla. Če le-to ni veljavno, zahtevo zavrže.

V drugem delu diplomske naloge je predstavljen programski del sistema Vratar. Poleg strežniških aplikacij je opisan tudi razvoj mobilne aplikacije mVratar – odjemalca za operacijski sistem iOS[1], kar je bil glavni cilj diplomskega dela. Navedene so lastnosti aplikacije ter težave, s katerimi smo se srečali pri samem razvoju.

Diplomsko delo se zaključi s povzetjem glavnih značilnosti sistema in predstavitvijo možnosti izboljšav ter idej za nadaljnji razvoj aplikacije.

[1]iOS (znan tudi kot iPhone OS) je Applov prenosni operacijski sistem.

Poglavje 2

Sistem Vratar – strojni del

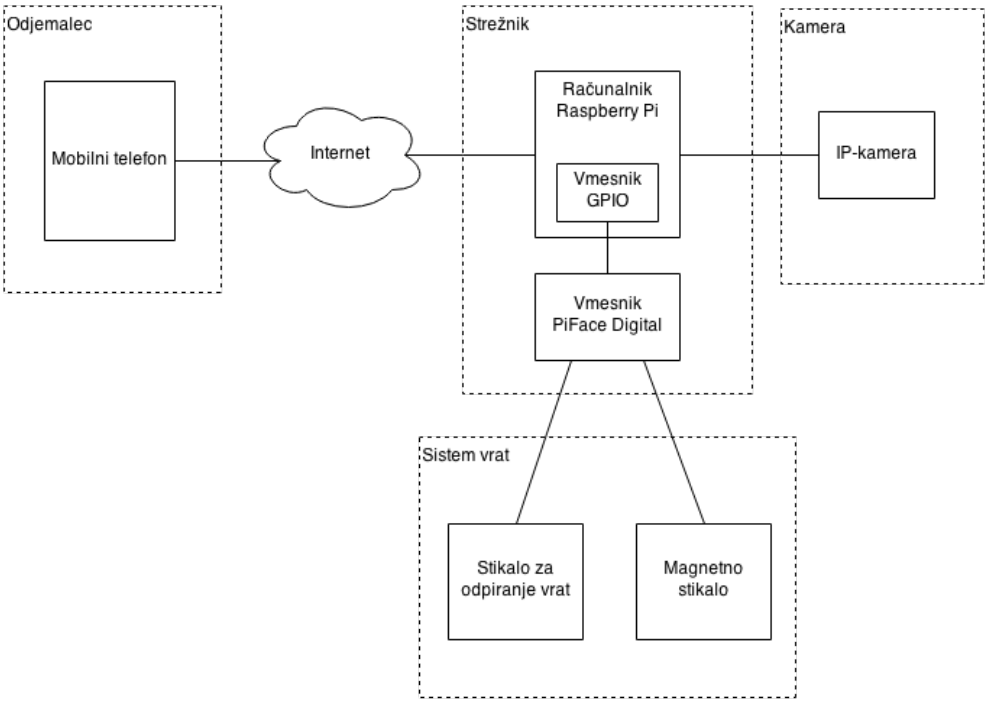
Sistem Vratar je namenjen krmiljenju garažnih vrat s pomočjo mobilne aplikacije. Za nadzor sistema se uporablja mobilni telefon, na katerem je nameščena aplikacija mVratar. Z aplikacijo se preko svetovnega spleta prijavimo na strežnik sistema Vratar. Po uspešni avtentikaciji nam aplikacija prikaže trenutno stanje vseh vrat, ki so vključena v sistem. Posamezna vrata lahko nato s preprosto gesto odpiramo ali zapiramo.

Razvili smo dve različici sistema Vratar. Pri obeh različicah smo strežniško aplikacijo namestili na računalnik Raspberry Pi. Glavna razlika med različicama je v načinu povezave strežnika s sistemom vrat, kjer se nahajajo senzorji in stikalo za krmiljenje vrat. V prvotni različici sistema smo povezave realizirali z uporabo žične povezave. Te smo v drugi različici sistema nadomestili z RF-moduli, ki med sabo komunicirajo preko radijskih valov. Obe različici imata tudi možnost zajema slike vrat s pomočjo mrežne kamere.

2.1 Žična različica sistema Vratar

Na sliki 2.1 je prikazana shema žične različice sistema Vratar. Kot smo že omenili, se za nadzor sistema uporablja aplikacija mVratar, ki deluje na mobilnih napravah z operacijskim sistemom iOS. Razvoj aplikacije je potekal v razvojnem okolju Xcode. Aplikacija nam poleg krmiljenja vrat nudi še nekatere druge funkcionalnosti, katerih podrobnejši opis najdemo v tretjem poglavju.

Aplikacija se preko svetovnega spleta poveže na strežnik sistema Vratar. Za



Slika 2.1: Shema žične različice sistema Vratar.

komunikacijo med strežnikom in odjemalcem se uporablja protokol WebSocket, ki nam omogoča komunikacijo v realnem času. Strežniški del sistema sestavlja obratno posredniški strežnik Nginx, ki skrbi za avtentikacijo odjemalca in posredovanje odjemalčevih zahtev na ciljni strežnik. Ta je implementiran z uporabo programske platforme Node.js, ki nam omogoča izvajanje JavaScript programov na strani strežnika. Celoten strežniški del je nameščen na računalniku Raspberry Pi.

Raspberry Pi je mini računalnik, na katerem najdemo poleg ostalih vmesnikov tudi vmesnik GPIO. Gre za splošno-namenski vmesnik, ki se uporablja za komunikacijo s strojno opremo. Nanj smo priključili vmesnik PiFace digital, ki vsebuje elektronske elemente, na katere je priključen sistem vrat. Za razvoj programa, s katerim smo krmilili vmesnik GPIO, smo uporabili programsko platformo Node.js.

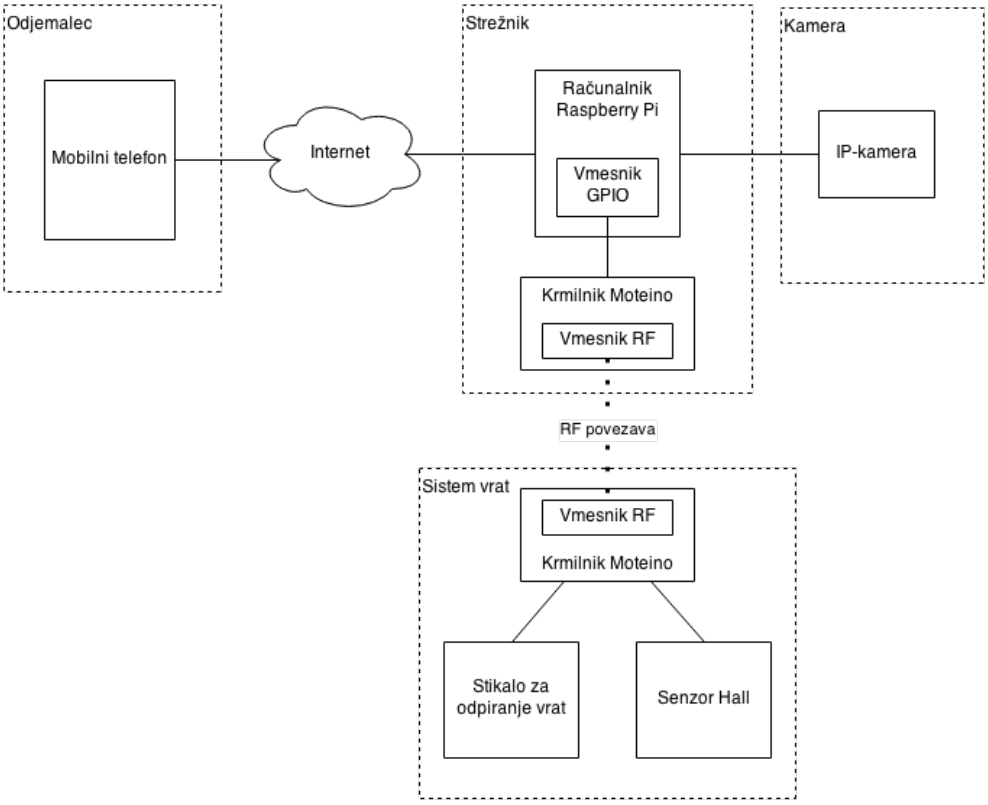
Vmesnik PiFace Digital vsebuje digitalne vhode, na katere smo priključili magnetna stikala. Ta nam služijo kot senzorji za zaznavanje trenutnega stanja vrat. Uporabili smo dve magnetni stikali, s pomočjo katerih lahko preverimo, če so vrata dosegla končno odprto oz. zaprto pozicijo. Namestili smo ju na okvir vrat, tako da se vključita ob prisotnosti magnetov, ki sta pritrjena na vrata.

Drugi element vmesnika PiFace Digital, ki smo ga uporabili za krmiljenje vrat, je rele. Bistvena lastnost tega elementa je galvanska ločitev, ki preprečuje vsakršen neposredni pretok električnega toka med sosednjima elektronskima vezjema. Rele smo priključili vzporedno s stikalom za krmiljenje vrat. S tem smo dosegli, da lahko stikalo vrat aktiviramo programsko.

2.2 Brezžična različica sistema Vratar

Brezžična različica sistema Vratar (slika 2.2) nam ponuja enako funkcionalnost kot prejšnja različica. Razlika je v tem, da pri tej različici nismo uporabili vmesnika PiFace Digital, ampak smo na vmesnik GPIO priključili ploščico Moteino. Gre za izpeljanko razvojne ploščice Arduino, ki temelji na Atmelovi družini mikrokrmilnikov in ima poleg ostalih vhodov in izhodov tudi RF (angl. Radio Frequency) oddajno-sprejemno enoto.

Ploščica Moteino, ki se nahaja na strežniku (v nadaljevanju bazni krmilnik), preko RF-povezave komunicira z drugo ploščico Moteino, ki je priključena na sistem vrat (v nadaljevanju podrejeni krmilnik). Podobno kot v žični različici sistema,



Slika 2.2: Shema brezžične različice sistema Vratar.

smo tudi tu za krmiljenje vrat uporabili rele, ki smo ga priključili na stikalo vrat. Krmilili smo ga s pomočjo enega izmed izhodov podrejenega krmilnika. Magnetna stikala smo nadomestili z manjšimi senzorji Hall, ki smo jih vezali na vhode podrejenega krmilnika. Tudi senzorji Hall se aktivirajo ob prisotnosti magneta, zato je bila namestitev senzorjev podobna kot pri žični različici.

Za bazni in podrejeni krmilnik je bilo potrebno razviti dva ločena programa. Razvoj je potekal v razvojnem okolju Arduino. Program baznega krmilnika skrbi za obdelavo ukazov, ki mu jih posreduje strežnik preko serijske povezave vmesnika GPIO. Iz ukaza razbere naslov podrejenega krmilnika, ki mu je ukaz namenjen. Ukazu nato doda časovno veljavno geslo in ga preko RF-povezave posreduje podrejenemu krmilniku.

Program podrejenega krmilnika bere podatke iz senzorjev in jih na zahtevo posreduje baznemu krmilniku. Poleg tega krmili rele za krmiljenje vrat. Pred izvršitvijo vsakega ukaza se preveri veljavnost časovno veljavnega gesla.

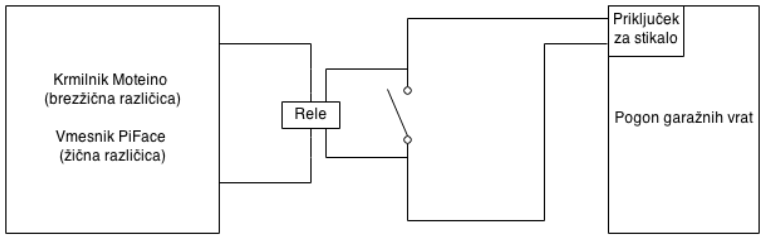
2.3 Uporabljena strojna oprema

V tem poglavju so na kratko opisani posamezni elementi sistema Vratar in njihova vloga v sistemu.

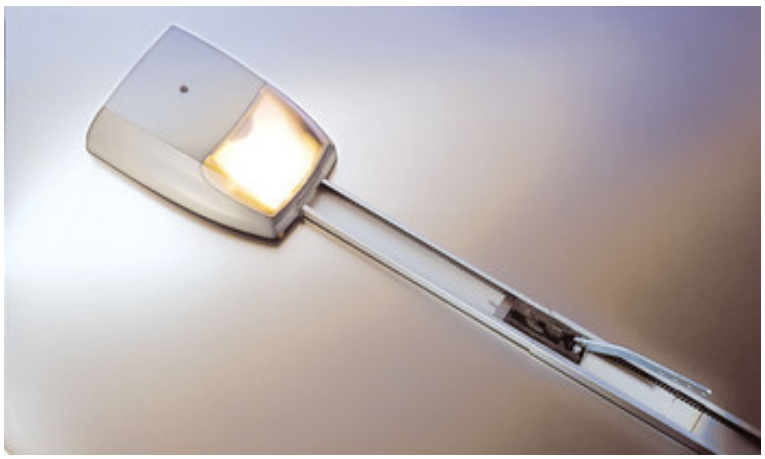
2.3.1 Pogon garažnih vrat

Splošno

Pogon garažnih vrat nam omogoča udobno odpiranje in zapiranje vrat z daljinskim upravljalnikom. Poleg tega je možno vrata krmiliti tudi z uporabo stikala, ki je povezano s pogonom garažnih vrat. V diplomski nalogi smo v sistem Vratar priključili dvoje vrat, opremljenih s pogonom znamke Crawford Ultra S (slika 2.4), ki je primeren za dvižna vrata do 50 kg [12]. Pogon je že tovarniško opremljen s sistemom za zaznavanje ovir, zato nam ni bilo potrebno nameščati dodatnih senzorjev. Če vrata med premikanjem naletijo na oviro, upravljalnik vrat samodejno zaustavi delovanje vrat in povzroči premikanje v nasprotno smer, dokler ne doseže ustreznega končnega položaja. Daljinski upravljalnik vrat je zaščiten s kodirno tehnologijo t. i. vrtečih se kod (angl. rolling code). Te se uporabljajo za



Slika 2.3: Povezava pogona vrat s sistemom Vratar.



Slika 2.4: Pogon garažnih vrat – Crawford Ultra S. Vir slike: [13]

preprečevanje napadov, kjer prisluškovalec posname radijski signal, ki ga oddaja daljinski upravljalnik pri odpiranju vrat, ter ga kasneje ponovno predvaja in s tem odpre vrata. Podobno zaščito radijskega signala smo implementirali v brezžični različici sistema Vratar.

Uporaba v sistemu Vratar

Povezava sistema Vratar s pogonom garažnih vrat je realizirana z uporabo releja, ki je vezan vzporedno s stikalom garažnih vrat (slika 2.3). S tem smo zagotovili, da sta sistema med seboj galvansko ločena.

Slika 2.5: Magnetno stikalo. Vir slike: [27]

Stikalo 1	Stikalo 2	Stanje vrat
0	0	Vrata se odpirajo ali zapirajo oz. so obtičala nekje vmes.
0	1	Vrata zaprta.
1	0	Vrata odprta.

Tabela 2.1: Stanja magnetnih stikal.

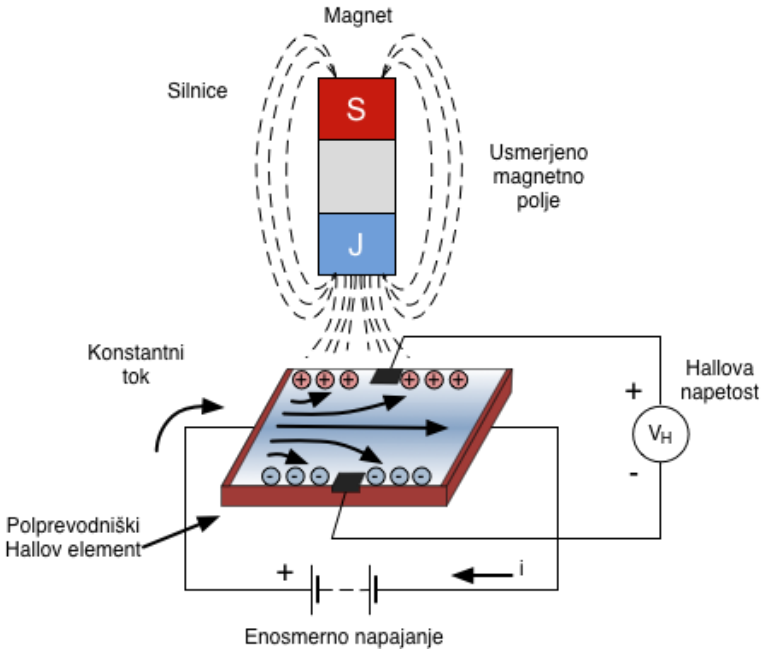
2.3.2 Magnetno stikalo

Splošno

Magnetno stikalo (slika 2.5) je sestavljeno iz dveh delov: magneta, ki je po navadi pritrjen na premikajoči se predmet (v našem primeru so to vrata) in stikala, ki se aktivira ob prisotnosti magneta.

Uporaba v sistemu Vratar

Magnetna stikala smo uporabili pri žični različici sistema Vratar, in sicer za preverjanje trenutnega stanja vrat. Stikalo smo vezali na vhod vmesnika PiFace Digital in ga pritrdili na okvir vrat. Ko se vrata zaprejo oz. odprejo, se magnet, ki je pritrjen na vrata, približa stikalu. Pri tem se stikalo sklene in na vhodu vmesnika PiFace Digital dobimo signal, kar pomeni, da so vrata odprta oz. zaprta. V tabeli 2.1 so prikazana vsa možna stanja vrat, ki jih dobimo na podlagi stanj magnetnih stikal. V brezžični različici smo ta stikala nadomestili s senzorji Hall.



Slika 2.6: Prikaz delovanja senzorja Hall.

2.3.3 Senzor Hall

Splošno

Senzor Hall [14, 17] je element, ki ob prisotnosti magneta z ustreznim magnetnim poljem generira določeno izhodno napetost, ki jo imenujemo Hallova napetost. Hallova napetost je odvisna od gostote magnetnega polja v okolici. Generirati se prične, ko gostota magnetnega pretoka preseže vnaprej določeno mejno vrednost. Gre za tako imenovan Hallov pojav, ki ga je odkril Edwin Hall leta 1879. Delovanje senzorja je prikazano na sliki 2.6.

Senzorji Hall so na voljo z analognim ali digitalnim izhodom. Pri analogni različici dobimo na izhodu Hallovo napetost, ki se povečuje oz. zmanjšuje v odvisnosti od moči magnetnega polja. V našem primeru smo uporabili digitalno različico, ki lahko ima na izhodu le dve možni stanji - 0 oz. 1. Senzor ima tri priključke, katerih katerih funkcija je opisana v tabeli 2.2.

Št. priključka	Oznaka	Funkcija
1	VDD	Napajalna napetost
2	OUT	Izhod
3	GND	Ozemljitev

Tabela 2.2: Opis priključkov senzorja Hall.

Uporaba v sistemu Vratar

Senzorje Hall smo uporabili za zaznavanje trenutnega stanja vrat v brezžični različici sistema. Uporabili smo dva senzorja, katera smo pritrdili na os pogona garažnih vrat. Poleg senzorjev je bilo potrebno namestiti še dva magneta. Pritrdili smo ju na jermen pogona in sicer tako, da je eden ob prvem senzorju, ko so vrata v končnem odprtem položaju, drugi pa ob drugem, ko so vrata zaprta. Slika 2.7 prikazuje položaj magnetov pri zaprtih vratih.

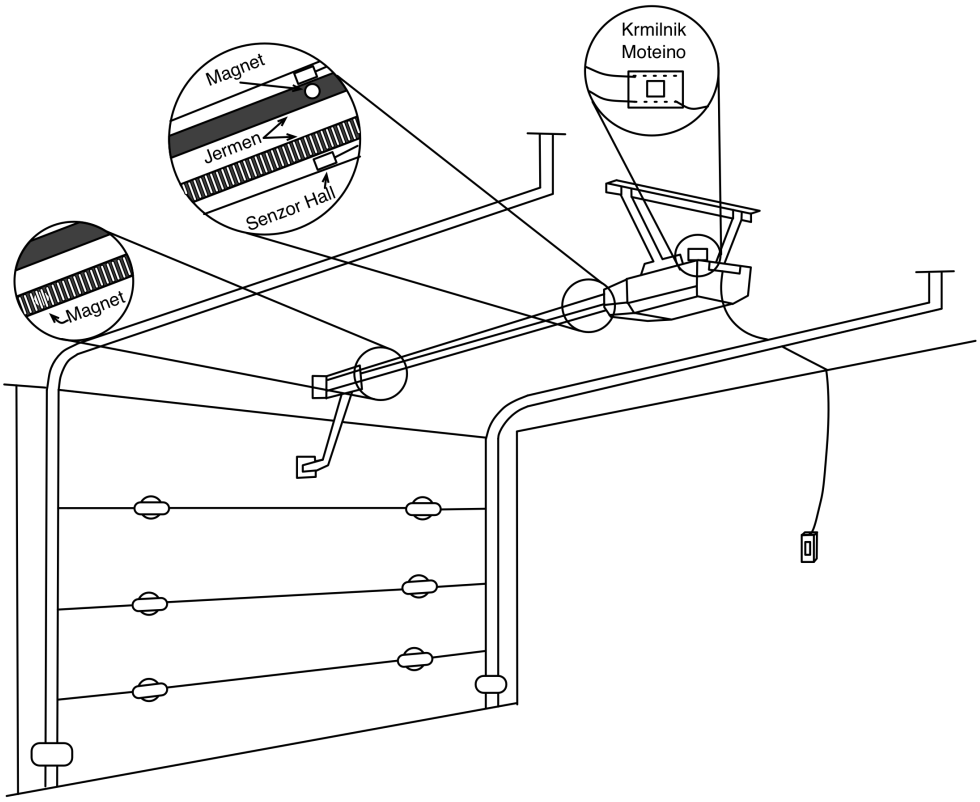
2.3.4 Računalnik Raspberry PI

Splošno

Raspberry Pi (slika 2.8) je računalnik nizkega cenovnega razreda, ki je nekoliko večji kot kreditna kartica. Razvila ga je neprofitna fundacija Raspberry Pi, da bi motivirala mlajše generacije, ki se podajajo v svet programiranja. Uporablja se kot manjši osebni računalnik za brskanje po spletu, kot medijski center za televizijske zaslone in seveda za zanimive projekte, ki nastanejo v glavah programerjev in ostalih ljubiteljev elektronike [15, 3].

Specifikacije

Računalnik Raspberry Pi je na voljo v dveh različicah, model A in model B. Močnejša različica B ima v primerjavi z modelom A dvakrat več pomnilnika, dvojni priključek USB in mrežni priključek. Vse to pa botruje nekoliko večji porabi električne energije, ki je v primerjavi z modelom B dvakrat večja (model A ima porabo 1.5 W). Za napajanje se uporablja vmesnik mikro USB. Sistem temelji na čipu Broadcom BCM2835. Gre za tako imenovan sistem na čipu (angl. SoC



Slika 2.7: Položaj magnetov pri zaprtih vratih.

Slika 2.8: Računalnik Raspberry Pi Model-B. Vir slike: [16]

	Raspberry Pi	
	Model A	Model B
Cena	25 $	35 $
Čip	Sistem na čipu Broadcom BCM2835	
CPE	ARM1176JZ-F 700 MHz	
GPE	Dvojedrni VideoCore IV® multimedijski koprocesor	
Pomnilnik RAM	256 MB SDRAM	512 MB SDRAM
Mrežni vmesnik	ne	10/100 Ethernet
Število USB 2.0 vmesnikov	1	2
Video izhodi	HDMI, RCA (PAL in NTSC)	
Audio izhodi	3.5 mm vtič, HDMI	
Pomnilniške kartice	SD,MMC,SDIO	
Operacijski sistem	Linux	
Dimenzije	8.6 cm x 5.4 cm x 1.5 cm	8.6 cm x 5.4 cm x 1.7 cm

Tabela 2.3: Primerjava modelov Raspberry Pi.

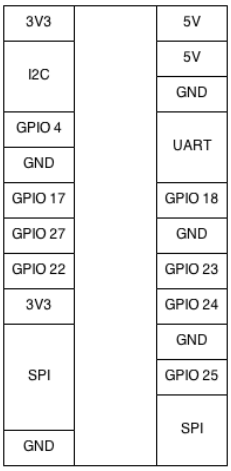
– System on chip) [35], kar pomeni, da so v čipu strnjeni vsi deli računalniškega sistema. Čip vsebuje 700 MHz procesor iz družine ARM11, grafično procesno enoto VideoCore IV in pomnilnik velikosti 256 oz. 512 MB. Operacijski sistem naložimo na dodatno pomnilniško kartico, katero vstavimo v ustrezno režo. Primerjava obeh modelov je prikazana v tabeli 2.3.

Vmesnik GPIO

GPIO [28] je splošno namenski vmesnik, na katerega lahko priključimo zunanjo strojno opremo. Sestavlja ga 26 priključkov različnih tipov (slika 2.9), in sicer:

- splošno namenski priključki
  Digitalni vhodi oz. izhodi, katere lahko uporabljamo npr. za prižiganje in ugašanje svetlečih diod.

- priključki vodila I^2C (angl. Inter-Integrated Circuit) [19]
  Vodilo omogoča priključitev strojne opreme, z uporabo le dveh priključkov



Slika 2.9: Priključki vmesnika GPIO.

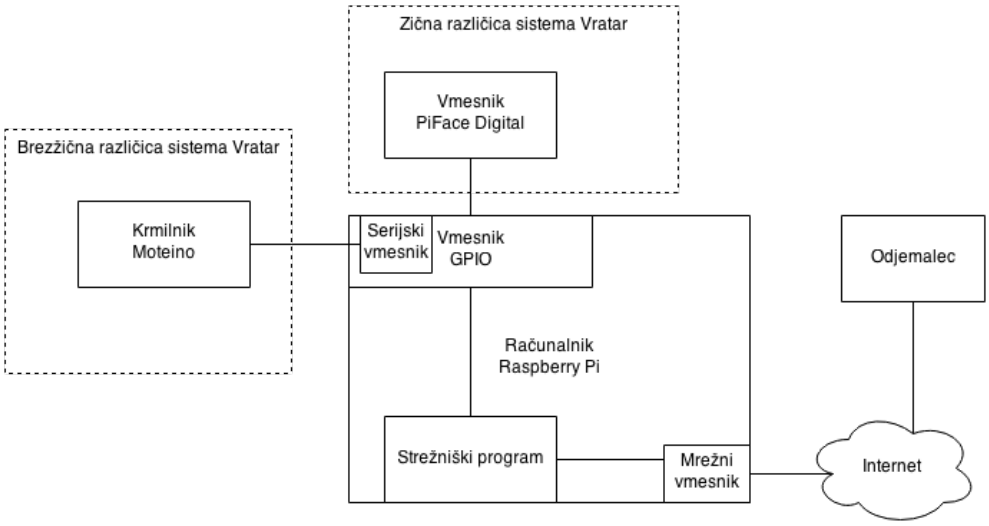
(SCL in SDA). SDA služi za prenos ukazov in podatkov. SCL je urin signal, ki sinhronizira prenos ter hkrati določa hitrost komunikacije. Vsaka naprava na vodilu ima svoj naslov in lahko deluje kot sprejemnik in/ali oddajnik.

- priključki vodila SPI (angl. Serial Peripheral Interface Bus) [34]
  Vodilo omogoča dvosmerno komunikacijo z eno ali več podrejenimi napravami. Pri tem uporablja štiri priključke (MISO, MOSI, SCKL in CE0 oz. CE1). Po liniji MISO pošilja podrejena naprava podatke nadrejeni napravi, pri MOSI pa poteka komunikacija v obratni smeri. SCKL je urin signal, ki ga določa nadrejena naprava. Preko CE0 oz. CE1 pa nadrejena naprava določi, s katero podrejeno napravo komunicira.

- priključki serijskega vmesnika
  Uporabljata se priključka Rx (sprejem) in Tx (oddaja).

Uporaba v sistemu Vratar

Na sliki 2.10 je prikazana vloga računalnika Raspberry Pi v sistemu Vratar. V žični različici sistema služi vmesnik GPIO kot digitalni vhod in izhod, pri brezžični različici pa smo ga uporabili kot serijski vmesnik.



Slika 2.10: Uporaba računalnika Raspberry Pi v sistemu Vratar.

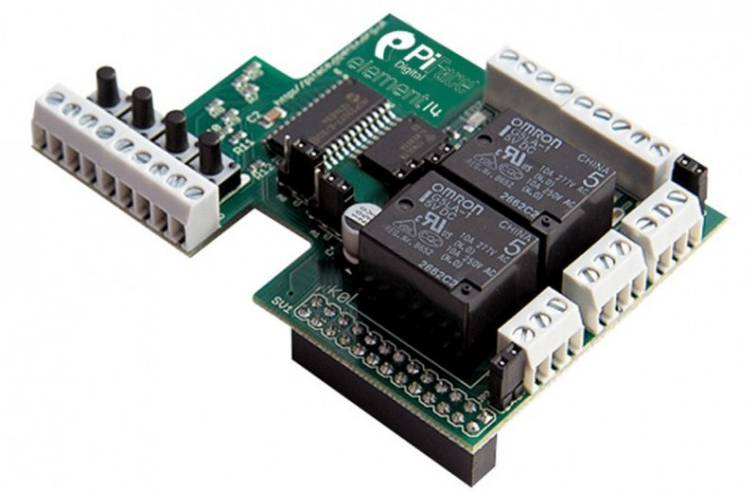
2.3.5 Vmesnik PiFace Digital

Splošno

Vmesnik PiFace Digital (slika 2.11) [31] nam omogoča, da na hiter in enostaven način povežemo računalnik Raspberry Pi z zunanjim svetom. Narejen je tako, da se popolnoma prilega obliki računalnika Raspberry Pi. Priključimo ga na vmesnik GPIO, preko katerega tudi upravljamo vsa nadaljnja dejanja. Interakcija z zunanjim svetom poteka preko elementov, ki so na vmesniku.

Značilnosti:

- priključitev neposredno na vmesnik GPIO računalnika Raspberry Pi,
- 2 releja,
- 4 stikala,
- 8 digitalnih vhodov,
- 8 izhodov z odprtim kolektorjem,
- 8 svetlečih diod,



Slika 2.11: Vmesnik PiFace Digital. Vir slike: [30]

- preprosto ga je programirati v programskih jezikih Python, Scratch in C,
- grafični emulator in simulator.

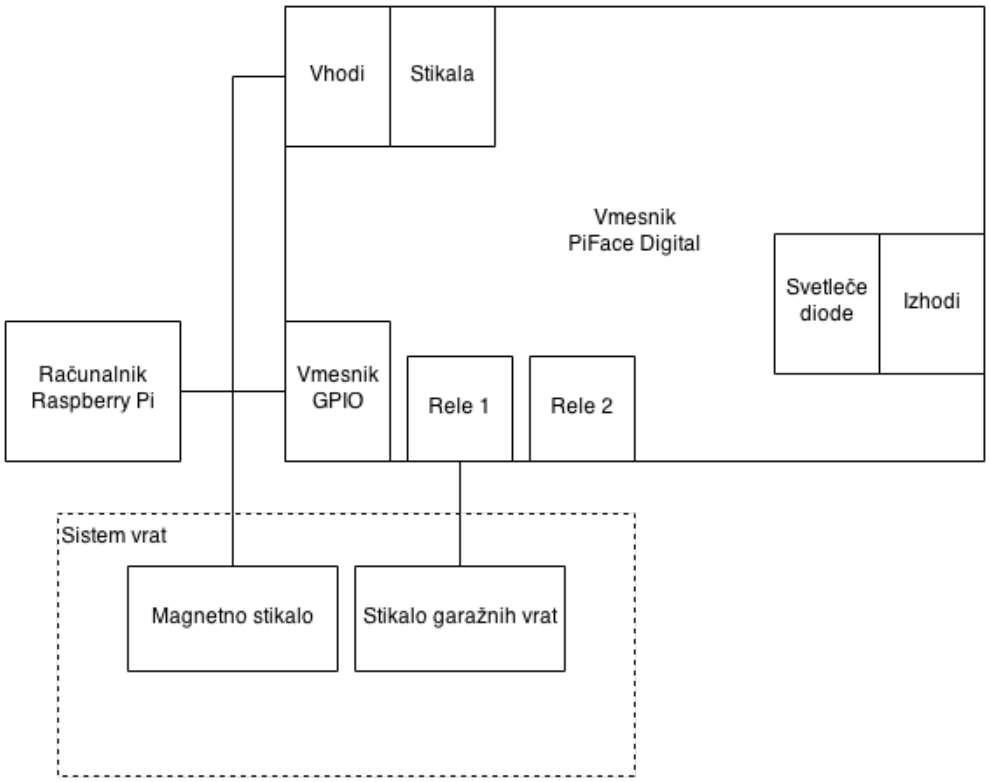
Uporaba v sistemu Vratar

V žični različici sistema Vratar smo posamezne elemente sistema vrat priključili na strežnik preko vmesnika PiFace Digital (slika 2.12). Za preverjanje stanja vrat smo uporabili digitalne vhode, na katere smo priključili magnetni stikali. Vrata pa smo krmilili z uporabo releja, katerega smo povezali vzporedno s stikalom garažnih vrat. S pomočjo releja smo dosegli galvansko ločitev našega sistema in stikala garažnih vrat.

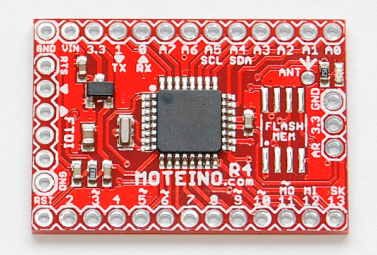
2.3.6 Ploščica Moteino (izpeljanka razvojne ploščice Arduino)

Splošno

Razvojna ploščica Arduino [26, 1] temelji na Atmelovi družini mikrokrmilnikov AVR. Za programiranje se uporablja poenostavljena različica programskega jezika C/C++. Nanj lahko priključimo različne senzorje, servo motorje in še veliko drugih



Slika 2.12: Uporaba vmesnika PiFace Digital v sistemu Vratar.



Slika 2.13: Ploščica Moteino. Vir slike: [20]

stvari. Na voljo je več različnih modelov ploščic. Med njimi sta najbolj popularni ploščici Arduino Uno in Leonardo. Posamezne ploščice se med sabo razlikujejo predvsem po številu vhodnih in izhodnih priključkov.

Shema tiskanega vezja ploščice je prosto dostopna in jo lahko poljubno spreminjamo ter uporabljamo. Zaradi tega se na trgu pojavlja poleg osnovnih modelov ploščice Arduino tudi veliko drugih izpeljank. V diplomski nalogi smo pri brezžični različici sistema uporabili izpeljanko, ki se imenuje Moteino (slika 2.13) [29]. Gre za poceni različico, ki temelji na čipu ATmega328 in je zato popolnoma združljiva s programskim okoljem Arduino. Ploščica Moteino nima priključka USB, zaradi tega je tudi nekoliko manjša in cenejša. Programiramo jo preko vmesnika FTDI, ki omogoča povezavo med serijskim vmesnikom ploščice in USB-vmesnikom računalnika. Glavni razlog za izbiro te izpeljanke ploščice Arduino je ta, da ima poleg ostalih vhodov in izhodov tudi RF-oddajno-sprejemno enoto, s pomočjo katere smo nadomestili žično povezavo med strežnikom in senzorji. Tabela 2.4 prikazuje specifikacije ploščice Moteino.

Izdelava ploščic Moteino

Ker se ploščice Moteino ne prodajajo v Evropi, smo se odločili, da si jih sestavimo sami. Pri tem smo potrebovali elemente za površinsko montažo (angl. SMD – surface-mount device), ki so prikazani v tabeli 2.5.

Poleg elementov smo potrebovali še tiskano vezje, katerega smo dali izdelati po načrtu. Elemente smo namestili na tiskano vezje in jih spajkali ročno s spajkalno postajo za SMD. Postopek spajkanja celotnega modula se je izkazal za dokaj zahtevno opravilo, ki zahteva veliko mero potrpežljivosti, natančnosti ter doslednosti.

Ploščica Moteino	
Mikrokrmilnik	ATmega328
Oddajno-sprejemna enota	RFM69 W/HW/CW
Frekvence oddajno-sprejemne enote	434 MHz (univerzalna), 868 MHz (Evropa), 915 MHz (ZDA, Avstralija itd.)
Delovna napetost	3.3 V
Napajalna napetost (priporočena)	3.3 V–9 V
Napajalna napetost (zgornja omejitev)	3.3 V–13 V
Digitalni V/I priključki	14 + 6 (6 s podporo PWM[1])
Analogni vhodni priključki	8
Enosmerni tok na V/I priključek	30 mA
Enosmerni tok za 3.3 V priključek	40 mA
Bliskovni pomnilnik	32 KB (1 KB je uporabljen za zagonski nalagalnik DualOptiboot)
SRAM	2 KB
EEPROM	2 KB
Ura procesorja	16 MHz

Tabela 2.4: Ploščica Moteino - specifikacije.

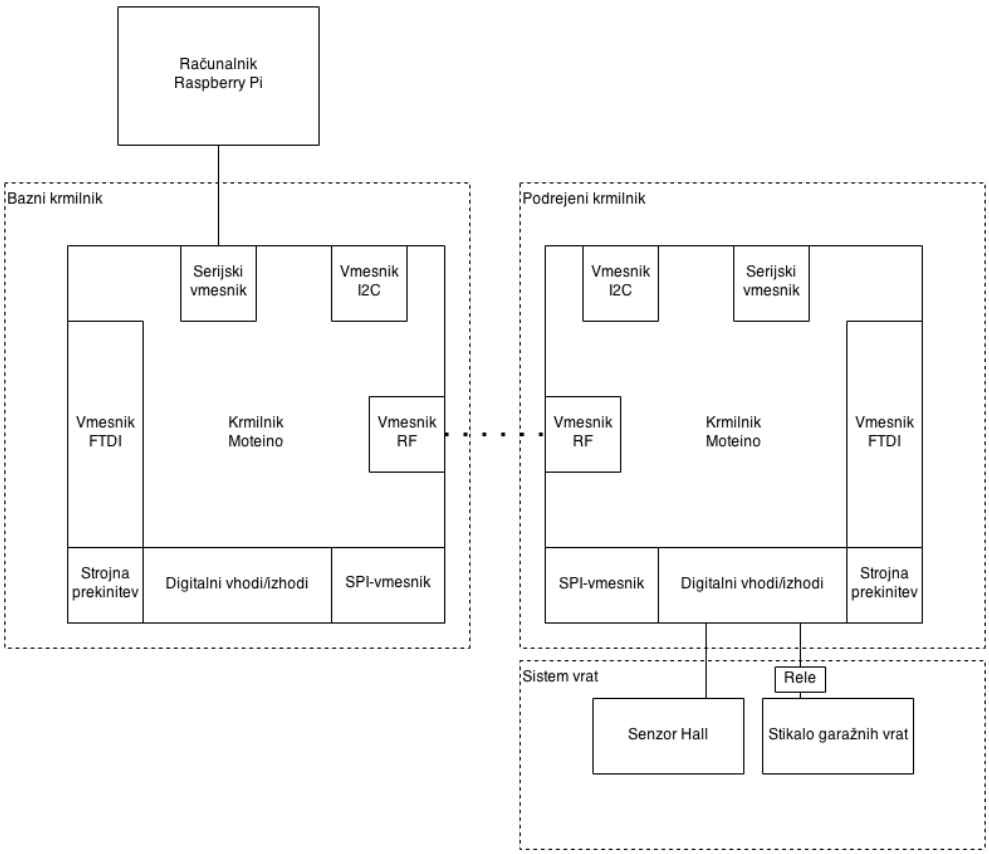
Element	Količina
Mikrokrmilnik ATmega328P	1
Keramični resonator 16 Mhz	1
Kondenzator 0.1 µf 0603	4
Kondenzator 10 µf	1
Upor 10 kΩ 0603	1
Napetostni regulator MCP1702 3v3	1
Modul RFM69HW 433Mhz	1
Rumena svetleča dioda LED 0603	1
Upor 1.5 kΩ 0603	1

Tabela 2.5: Ploščica Moteino – seznam elementov za površinsko montažo.

Pred uporabo modula je bilo potrebno na mikrokontrolerje ATmega328P naložiti zagonski program (angl. bootloader). To smo naredili s pomočjo programatorja Atmel AVRISP mkII.

Uporaba v sistemu Vratar

Slika 2.14 prikazuje uporabo dveh ploščic Moteino (v nadaljevanju bazni in podrejeni krmilnik) v brezžični različici sistema Vratar. Za razliko od žične različice je tu sistem vrat popolnoma ločen od strežnika. Bazni krmilnik, ki se nahaja na strani strežnika, je preko serijskega vmesnika povezan z računalnikom Raspberry Pi. Njegova naloga je, da sprejema ukaze strežnika in jih posreduje podrejenemu krmilniku, ki se nahaja na strani sistema vrat. Podrejeni krmilnik izvrši prejeti ukaz in posreduje povratno informacijo baznemu krmilniku, ta pa naprej strežniku. Komunikacija med baznim in podrejenim krmilnikom poteka preko radijske povezave.



Slika 2.14: Uporaba ploščice Moteino v sistemu Vratar.

2.3.7 Prikazovalnik LCD

Splošno

"Hitachi HD44780 LCD" je eden izmed najbolj pogostih prikazovalnikov LCD, ki se pojavljajo na tržišču. Gre za znakovni štirivrstični prikazovalnik LCD, ki ima vgrajeno krmilno logiko. V tabeli 2.6 so opisani priključki, preko katerih krmilimo prikazovalnik LCD [18, 4] .

Značilnosti:

- 20 znakov, 4 vrstice,
- belo besedilo na modrem ozadju,
- preprosta priključitev na testno ploščo,
- osvetlitev ozadja,
- krmiljenje je možno z uporabo le šestih digitalnih linij,
- vgrajena podpora za angleški/japonski nabor znakov,
- podpora za 8 dodatnih znakov.

Uporaba v sistemu Vratar

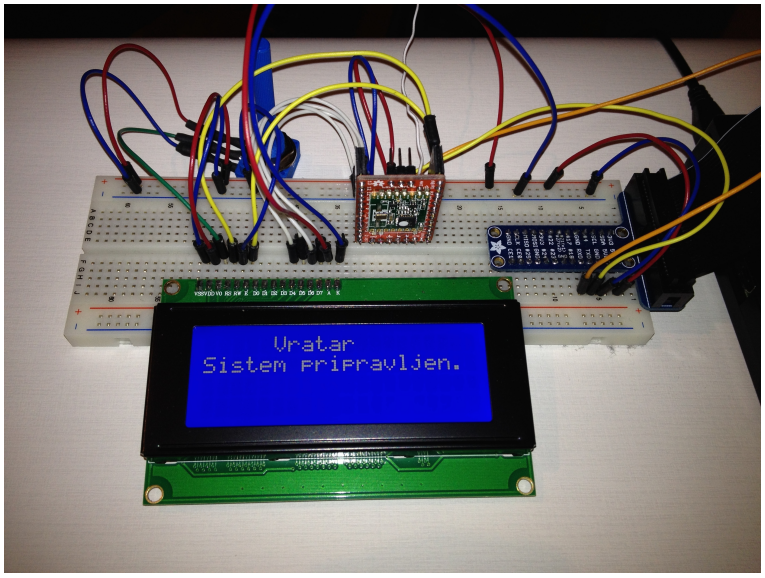
Prikazovalnik LCD (slika 2.15) smo uporabili v testni fazi brezžične različice sistema Vratar, in sicer pri testiranju aplikacije za ploščico Moteino. Nanj smo izpisovali sporočila in ukaze, ki se prenašajo preko brezžične komunikacije med posameznimi ploščicami Moteino.

2.3.8 Brezprekinitveno napajanje

Brezprekinitveno napajanje nam zagotavlja, da naš sistem deluje tudi v primeru izpada električne energije. Uporabili smo model za domačo uporabo "APC Back-UPS 500", ki nam nudi 500 VA / 300 W izhodne moči. [5].

Št. priključka	Oznaka	Funkcija
1	V_{SS}	Ozemljitev
2	V_{DD}	Napajalna napetost (od 3.3 V do 5 V)
3	V_0	Nastavitev kontrasta
4	RS	Izbira registra (0: ukazni, 1: podatkovni)
5	R/W	Branje/pisanje (0: pisanje, 1: branje)
6	E	Vključitev ure
7	DB0	Podatkovni bit 0 (ni v uporabi pri 4-bitnih operacijah)
8	DB1	Podatkovni bit 1 (ni v uporabi pri 4-bitnih operacijah)
9	DB2	Podatkovni bit 2 (ni v uporabi pri 4-bitnih operacijah)
10	DB3	Podatkovni bit 3 (ni v uporabi pri 4-bitnih operacijah)
11	DB4	Podatkovni bit 4
12	DB5	Podatkovni bit 5
13	DB6	Podatkovni bit 6
14	DB7	Podatkovni bit 7
15	LEDA	Osvetlitev – anoda
16	LEDK	Osvetlitev – katoda

Tabela 2.6: Opis priključkov prikazovalnika LCD.



Slika 2.15: Prikazovalnik LCD.

2.3.9 Mrežna kamera

Za vizualno preverjanje stanja vrat smo uporabili mrežno kamero proizvajalca Axis, model 213 PTZ (Slika 2.16). Uporaba kamere pride v poštev predvsem pri krmiljenju vrat iz oddaljene lokacije, kjer nimamo pogleda na vrata. Mobilna aplikacija mVratar nam po izvršitvi akcije za krmiljenje vrat samodejno prikaže sliko vrat, ki jo pridobi iz kamere.

Slika 2.16: Mrežna kamera Axis 213 PTZ. Vir slike: [10]

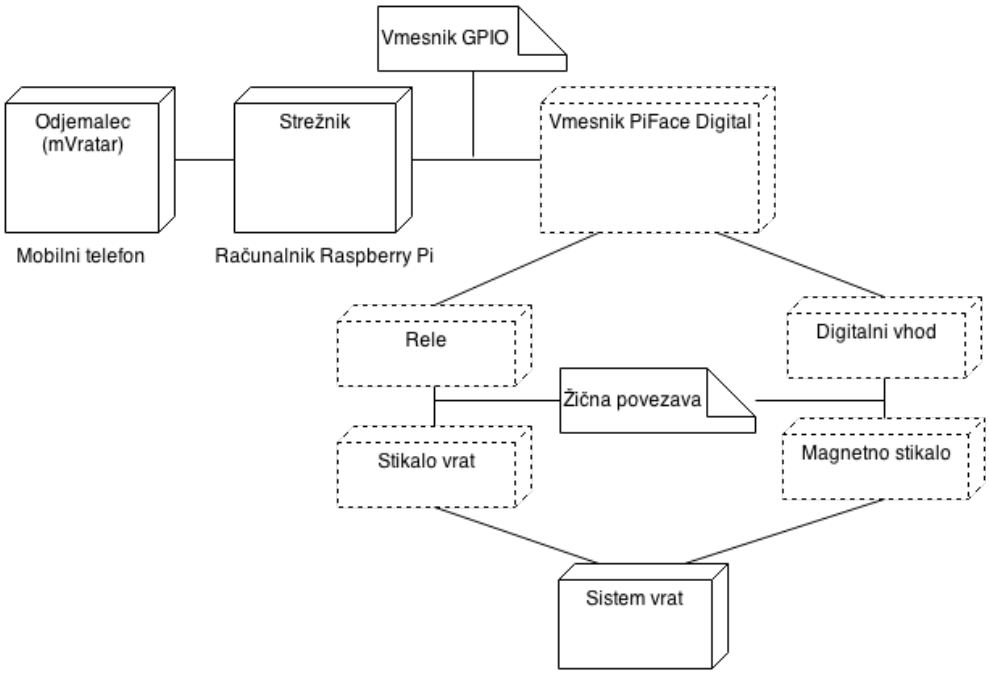
Poglavje 3

Sistem Vratar – programski del

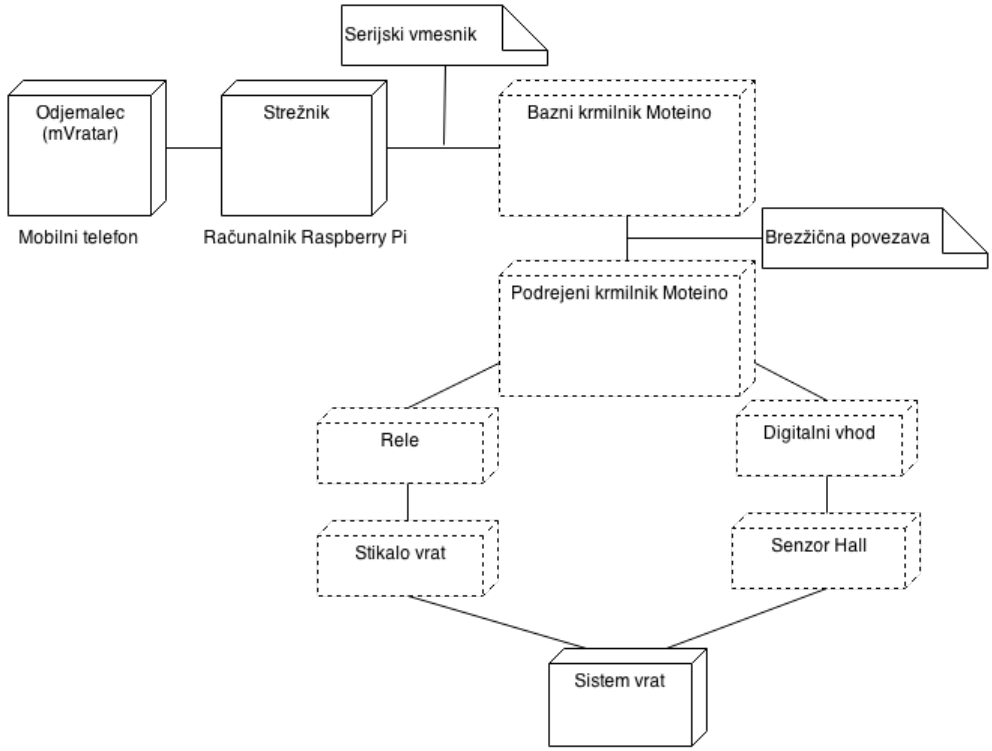
V tem poglavju je predstavljen programski del sistema Vratar. Povezave med posameznimi deli sistema so prikazane na slikah 3.1 in 3.2. Odjemalca predstavlja mobilna aplikacija mVratar, ki preko svetovnega spleta pošilja ukaze strežniku. Naloga strežnika je, da skrbi za avtentikacijo odjemalca in izvrševanje njegovih ukazov. Pri tem preko ustreznega vmesnika krmili sistem vrat. V žični različici sistema poteka krmiljenje vrat preko vmesnika PiFace Digital, na katerem so digitalni vhodi in rele, ki je z žično povezavo povezan s sistemom vrat. V brezžični različici sistema strežnik preko serijske povezave posreduje ukaze baznemu krmilniku Moteino. Naloga baznega krmilnika je, da ukazom doda časovno omejeno geslo in jih preko radijske povezave posreduje podrejenemu krmilniku, ki je na sistemu vrat. Podrejeni krmilnik skrbi za kontrolo pravilnosti gesla in ukazov ter za krmiljenje vrat.

Diagrama na slikah 3.3 in 3.4 predstavljata arhitekturo žične in brezžične različice sistema Vratar. Na diagramih so prikazane posamezne komponente, ki jih sistem potrebuje za svoje delovanje, in povezava med njimi.

Seznam dogodkov, ki se zgodijo pri prijavi odjemalca na strežnik, je sledeč. Odjemalec (aplikacija mVratar) vzpostavi povezavo z obratno-posredniškim strežnikom Nginx. Ta del smo implementirali z uporabo knjižnice `socket.IO`. Pri vzpostavitvi povezave se najprej izvede postopek avtentikacije. Ko je odjemalec avtenticiran, pošlje ukaz za pridobitev informacije o trenutnem stanju vrat. Komunikacija poteka preko protokola WebSocket. Povratno posredniški strežnik Nginx preusmeri



Slika 3.1: Splošni diagram žične različice sistema Vratar.



Slika 3.2: Splošni diagram brezžične različice sistema Vratar.

ukaz na spletni strežnik, ki smo ga implementirali z uporabo programske platforme Node.js.

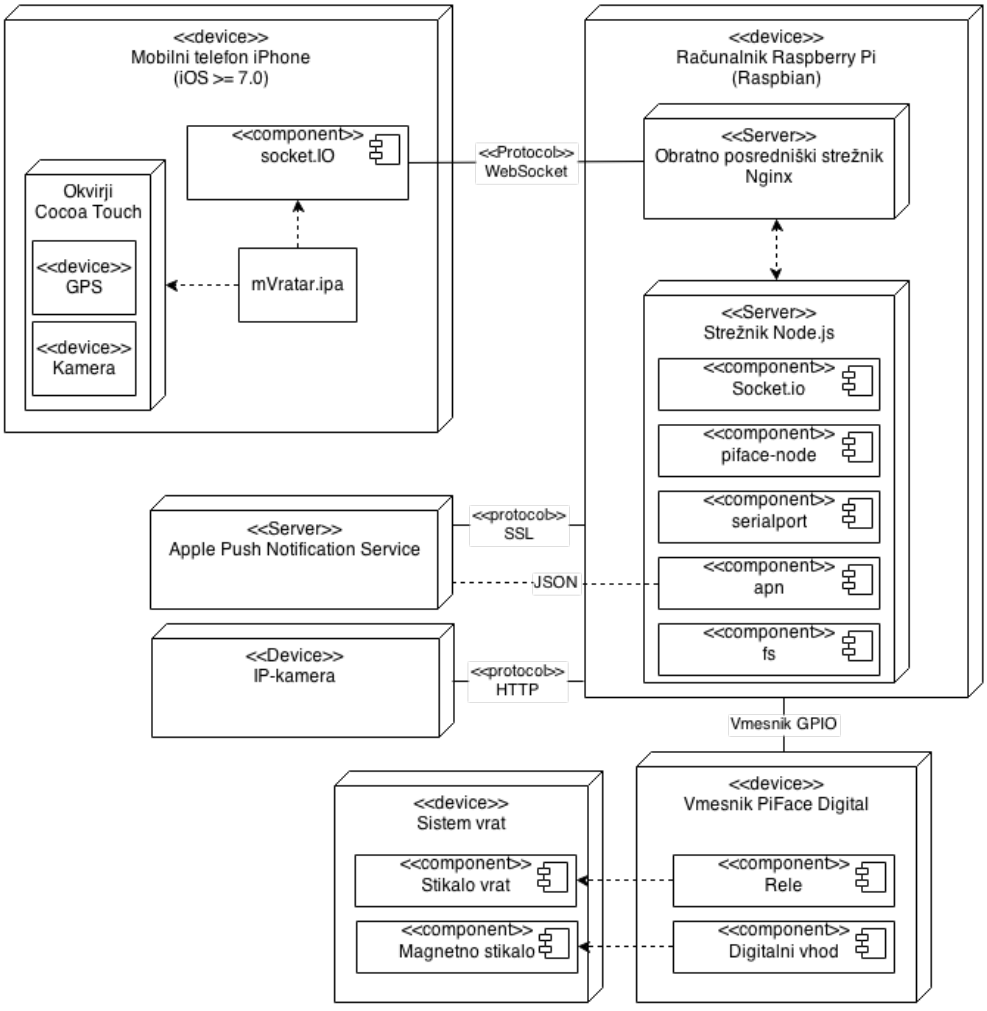
V žični različici sistema Vratar se informacija o trenutnem stanju vrat pridobi iz vmesnika PiFace Digital. Tu je bil uporabljen modul `piface-node`, s pomočjo katerega se pridobi informacija iz digitalnega vhoda, ki je povezan z magnetnim stikalom. Pridobljena informacija se nato v obratni smeri posreduje odjemalcu.

V brezžični različici sistema Vratar pa se ukaz iz spletnega strežnika preko serijske povezave posreduje baznemu krmilniku. Za vzpostavitev serijske povezave smo uporabili modul `SerialPort`, na strani krmilnika pa knjižnico `SPI`. Na baznem krmilniku se izvaja skica `VratarBazni.ino`, ki najprej preveri veljavnost ukaza. Če je le-ta veljaven, se mu doda geslo, ki ima časovno omejeno veljavnost. Ukaz se nato posreduje preko šifriranega radijskega signala podrejenemu krmilniku, na katerem se izvaja skica `VratarPodrejeni.ino`. Ta ukaz sprejme in odgovori s potrdilom o uspešnem sprejemu. Če bazni krmilnik v določenem času ne dobi potrdila, ponovi pošiljanje (število ponovitev je omejeno). Za radijsko komunikacijo med krmilnikoma se uporablja knjižnica `RFM69`. Podrejeni krmilnik po končanem sprejemu preveri veljavnost ukaza in pravilnost časovno veljavnega gesla. Če sta oba pogoja izpolnjena, izvrši ukaz za branje informacije o stanju vrat in pošlje povratno informacijo baznemu krmilniku. Bazni krmilnik nato v obratni smeri posreduje informacijo odjemalcu.

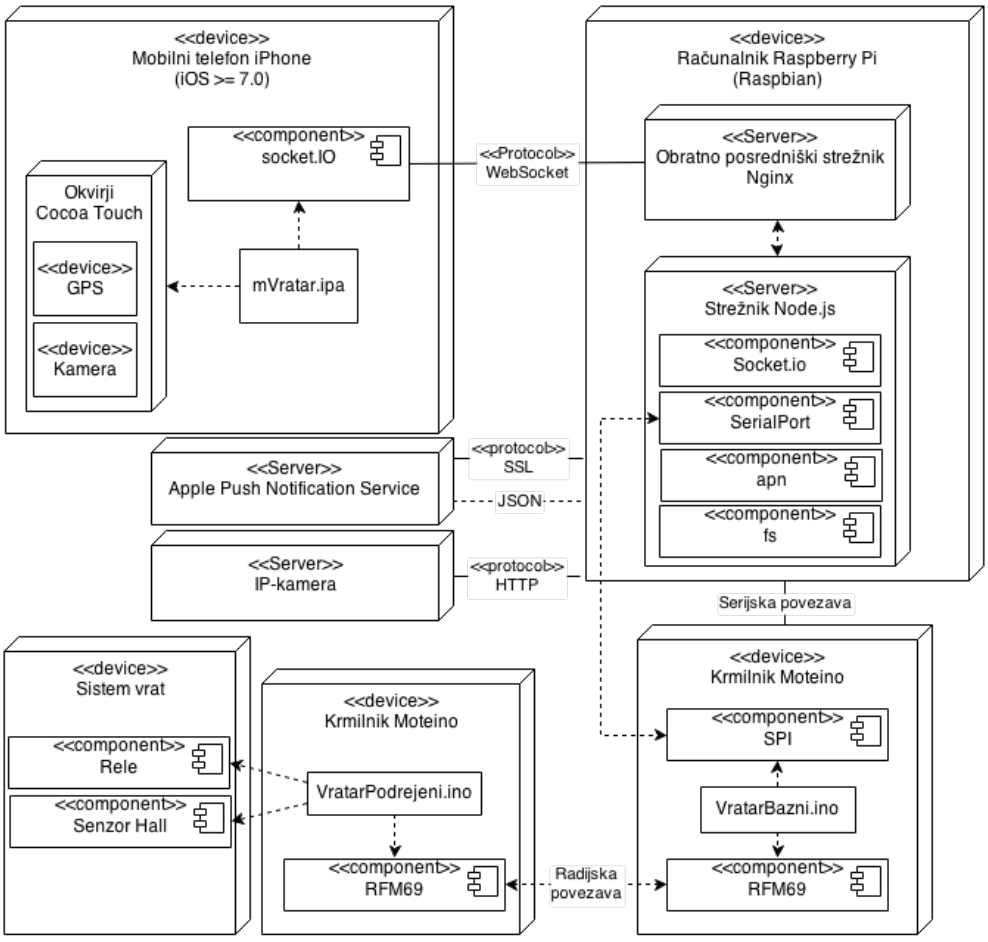
V nadaljevanju sledi opis posameznih programskih komponent.

3.1 Odjemalec - aplikacija mVratar

Za povezavo s strežnikom sistema Vratar smo razvili odjemalca za operacijski sistem iOS. Razvoj aplikacije, ki smo jo poimenovali mVratar, je potekal v razvojnem okolju Xcode. Aplikacija mVratar je preprosta, uporabniku prijazna mobilna aplikacija, ki omogoča intuitiven način krmiljenja vrat. V nadaljevanju poglavja je predstavljena funkcionalnost in uporaba aplikacije.



Slika 3.3: Diagram žične različice sistema Vratar.



Slika 3.4: Diagram brezžične različice sistema Vratar.

3.1.1 Pogoji za delovanje

Aplikacija mVratar deluje na mobilnih napravah z operacijskim sistemom iOS. Za delovanje je potrebna verzija sistema 7 ali več. Ker sistem Vratar temelji na principu strežnik – odjemalec, je za delovanje nujno potrebna povezava v domače omrežje. Pred uporabo aplikacije mVratar, je potrebno poskrbeti za nastavitve na strani strežnika. Poleg avtentikacijskih podatkov in ostalih nastavitev je potrebno nastaviti tudi enolične identifikatorje vrat, ki so priključena v sistem Vratar. Za aplikacijo mVratar je to ključni podatek, s katerim določamo, katera vrata želimo krmiliti. Identifikator vrat vnesemo poleg ostalih podatkov pri postopku dodajanja novih vrat.

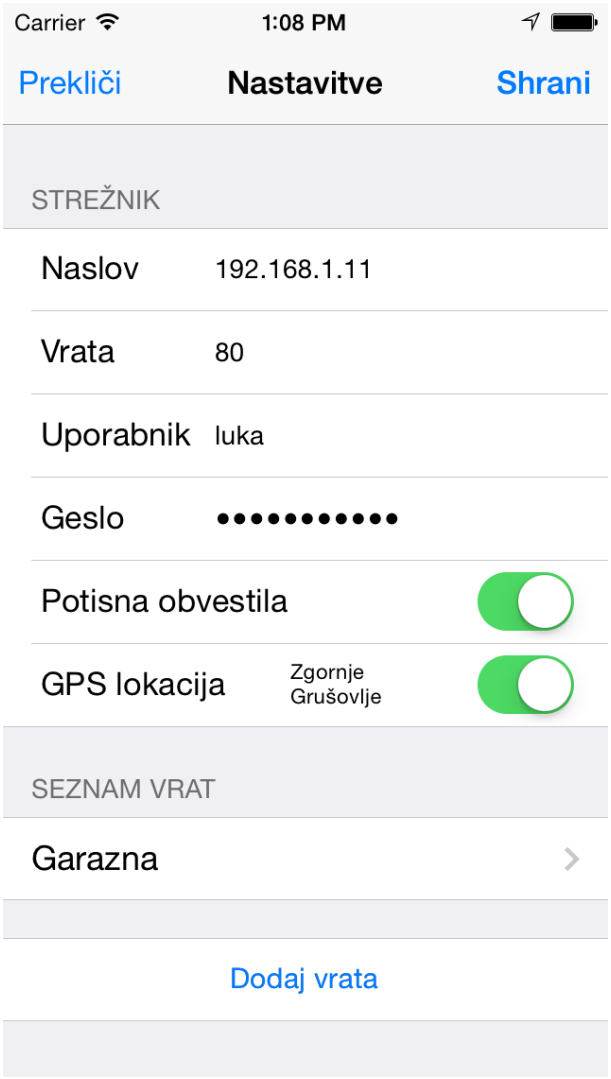
3.1.2 Funkcionalnosti aplikacije mVratar

Povezovanje na strežnik

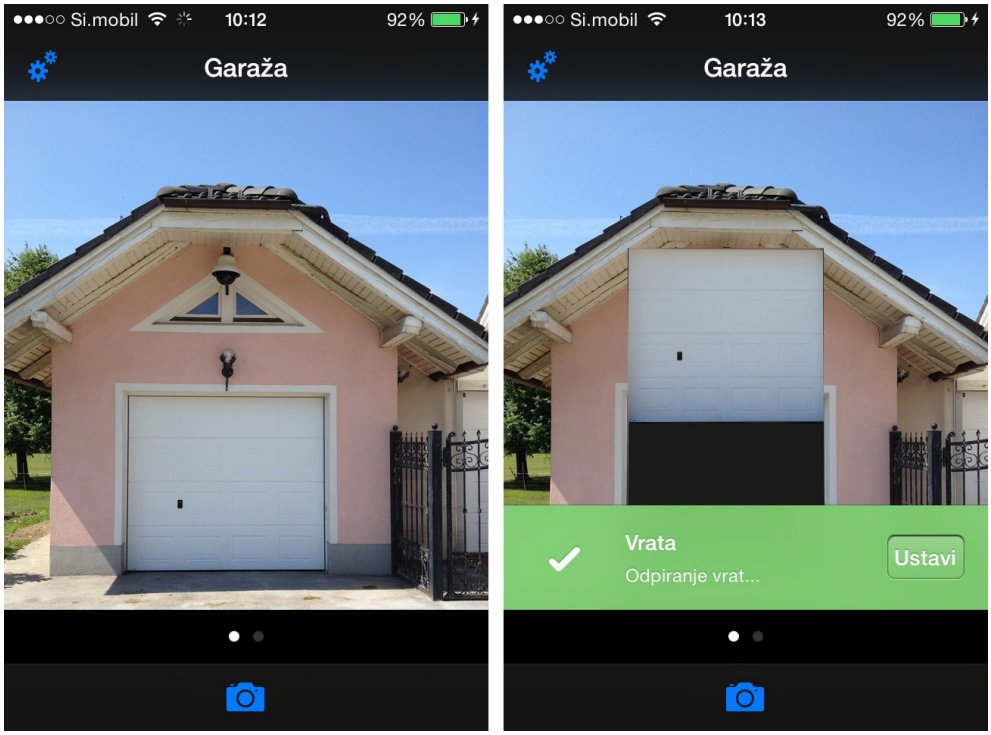
Ob prvem zagonu aplikacije Vratar je potrebno dodati nastavitve strežnika in seznam vrat, ki jih želimo krmiliti. Ob vsakem naslednjem zagonu aplikacije se povezava na strežnik vzpostavi samodejno. Za komunikacijo s strežnikom smo uporabili knjižnico Socket.IO, ki omogoča povezavo preko protokola WebSocket. V primeru 3.1 je prikazana metoda, ki se uporablja pri povezovanju na strežnik.

Primer 3.1: Metoda, ki se uporablja za povezavo na strežnik sistema Vratar.

```
/** Povezava na socket streznik */
- (void)connect
{
    if (!_socketIO.isConnecting && //povezovanje je ze v teku
        [self.server isValid] ) { //na voljo so vsi podatki, ki so potrebni
            za povezavo na streznik
        [_socketIO disconnect];
        [_socketIO connectToHost:self.server.host
                          onPort:self.server.port
                    withUsername:self.server.username
                     andPassword:self.server.password];
    }
}
```



Slika 3.5: Nastavitve aplikacije Vratar.



Slika 3.6: Vmesnik aplikacije Vratar.

Krmiljenje vrat

S pomočjo aplikacije mVratar lahko krmilimo poljubno število vrat (zgornja meja je odvisna od omejitev strežnika). To je tudi eno izmed dejstev, ki jih je bilo potrebno upoštevati pri razvoju vmesnika. Pomembno je namreč, da uporabnik kar se da hitro ugotovi, katera vrata predstavljajo trenutni pogled. Rešitve, kot so npr. napis poleg gumba, se nam v tem primeru niso zdele primerne, zato smo poiskali drugo možnost. Uporabniku smo omogočili, da s kamero telefona zajame sliko objekta z vrati, ki jih želi krmiliti. Na sliki nato določi območje, kjer se nahajajo vrata. Izbrano območje je predstavljeno z drsnim stikalom, ki se uporablja za krmiljenje vrat, hkrati pa prikazuje trenutno stanje vrat. Primer 3.2 prikazuje del programske kode, ki se kliče pri preklapljanju drsnega stikala. Po seznamu vrat se prestavljamo s potegom v levo oz. desno. Vmesnik aplikacije Vratar je prikazan na sliki 3.6.

Primer 3.2: Metode, ki se prožijo pri preklapljanju stikala vrat.

```
//LHPageContentViewController.m

- (void)toggleDoor
{
    LHDoor *door = [[[LHDoorManager sharedDoorManager] doors] objectAtIndex:
        self.pageIndex];
    [[LHServerManager sharedServerManager] toggleDoorWithId:door.doorId];
}

#pragma mark - LHSlideSwitchProtocol

- (void)slideSwitchDidOpen:(LHSlideSwitchView *)slideSwitchView
{
    [self toggleDoor]; //odpremo vrata
}

- (void)slideSwitchDidClose:(LHSlideSwitchView *)slideSwitchView
{
    [self toggleDoor]; //zapremo vrata
}

//LHServerManager.m
- (void)toggleDoorWithId:(NSInteger)doorId
{
    [_socketIO sendEvent:kCmdDoorToggle withData:[NSNumber numberWithInteger
        :doorId]];
}
```

Odpiranje vrat z uporabo lokacijskih storitev

Aplikacija mVratar omogoča dva načina odpiranja vrat. Poleg odpiranja z uporabo drsnega stikala, se lahko odpiranje sproži tudi samodejno, ko se odjemalec iz oddaljene lokacije približa lokaciji strežnika. Lokacijo strežnika določimo v nastavitvah aplikacije z izbiro opcije “GPS-lokacija”. Ta sproži postopek pridobitve lokacije z uporabo sistema za pozicioniranje (angl. GPS). Ko je lokacija strežnika določena, lahko funkcionalnost odpiranja na podlagi lokacije aktiviramo s pritiskom na gumb v orodni vrstici. Ta način odpiranja vrat je primeren predvsem za uporabo v vozilu, saj lahko funkcionalnost aktiviramo še preden se usedemo za volan. Pri vstopu v območje strežnika (tj. lokacija strežnika z radijem 200 m) se samodejno sproži ukaz za odpiranje vrat, funkcionalnost odpiranja na podlagi lokacije, pa se izključi. Če želimo uporabljati to funkcionalnost, mora mobilna

naprava podpirati lokacijske storitve. Poleg tega mora biti omogočen oddaljen dostop do sistema Vratar.

Metode, ki smo jih potrebovali za realizacijo te funkcionalnosti, so nam na voljo v okvirju `CoreLocation.framework`. Uporabljene so bile predvsem naslednje metode:

- `- (void)locationManager:(CLLocationManager *)manager didUpdateLocations:(NSArray *)locations`

  Metodo smo uporabili pri določitvi lokacije strežnika. Sproži se, ko so podatki o lokaciji na voljo.

- `- (void)locationManager:(CLLocationManager *)manager didEnterRegion:(CLRegion *)region`

  Metoda se sproži pri vstopu v eno izmed registriranih območij. V primeru, da to območje pripada strežniku sistema Vratar, se izvrši klic metode za odpiranje vrat.

- `- (void)startMonitoringForRegion:(CLRegion *)region`

  Metoda razreda `CLLocationManager`, ki vključi spremljanje podanega območja. Metoda se kliče ob pritisku na gumb za vključitev funkcionalnosti odpiranja vrat na podlagi lokacije.

- `- (void)stopMonitoringForRegion:(CLRegion *)region`

  Metoda razreda `CLLocationManager`, ki ustavi spremljanje podanega območja.

Potisna obvestila

Potisna obvestila omogočajo strežniku obveščanje odjemalca o določenih dogodkih v realnem času. Dobra lastnost potisnih obvestil je ta, da za prikaz obvestila ni potrebno imeti odprte aplikacije, saj se obvestilo prikaže sredi kateregakoli opravila na napravi (slika 3.7). Poleg besedila lahko vsebuje tudi zvočno obvestilo in značko, ki se prikaže na ikoni aplikacije. Za sprejem potisnih obvestil je potrebno v nastavitvah telefona vključiti dovoljenje sprejema obvestil za aplikacijo mVratar. Pri zagonu aplikacije se naprava preko storitve APNs prijavi na prejemanje določenih tipov obvestil. Pri tem prejme odgovor z naslovom naprave (angl. device token), ki ga nato posreduje strežniku sistema Vratar. Ta si naslove naprav

hrani v datoteki. Pošiljanje potisnih obvestil se sproži, ko krmilnik pošlje strežniku obvestilo o dogodku (npr. predolgo odprta vrata). Primer 3.3 prikazuje kodo, ki se uporablja za pridobitev žetona naprave.

Primer 3.3: Potisna obvestila - pridobitev žetona.

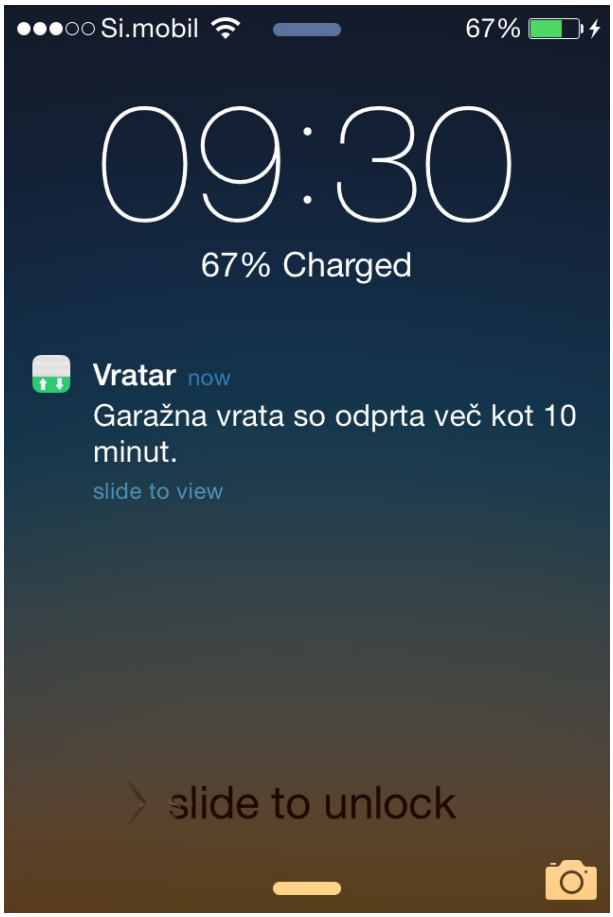
```
//LHAppDelegate.h
- (BOOL)application:(UIApplication *)application
    didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    //vkljucitev potisnih obvestil
        [[UIApplication sharedApplication]
            registerForRemoteNotificationTypes:(
            UIRemoteNotificationTypeSound | UIRemoteNotificationTypeAlert)];

    return YES;
}
...
- (void)application:(UIApplication*)application
    didRegisterForRemoteNotificationsWithDeviceToken:(NSData*)deviceToken
{
    NSString* token = [deviceToken description];
    [[LHServerManager sharedServerManager] registerDeviceToken:token];
}
```

Zvočna obvestila

Zvočna obvestila se uporabljajo kot povratne informacije, ki so posledica naslednjih dogodkov:

- Povezava s strežnikom je bila vzpostavljena.
- Povezava s strežnikom je bila prekinjena.
- Prišlo je do napake.
- Sprožili smo krmiljenje vrat.
- Vrata so se odprla.
- Vrata so se zaprla.
- Prihod v območje strežnika (odpiranje vrat na podlagi lokacije).

Za predvajanje zvočnih obvestil smo uporabili metodo `AudioServicesPlaySystemSound()`, ki jo najdemo v Cocoa Touch okvirju `AudioToolbox.framework`.



Slika 3.7: Primer potisnega obvestila sistema Vratar.

Kamera

Sistem Vratar ima možnost zajema slike iz mrežne kamere. Ta funkcionalnost služi kot dodatna kontrola pri preverjanju stanja vrat. V poštev pride predvsem pri krmiljenju vrat iz oddaljene lokacije. Sliko zajamemo s pritiskom na gumb v orodni vrstici. Možno je vključiti tudi samodejni prikaz slike, ki se prikaže v pojavnem oknu po prehodu vrat iz zaprtega v odprto stanje in obratno. Za prikaz pojavnega okna s sliko, se uporablja metoda iz primera 3.4.

Primer 3.4: Prikaz pojavnega okna s sliko, zajeto iz kamere.

```
- (void)showDoorImage
{
    [_doorManager getImageOfDoorWithId:(NSInteger)doorId image:^(UIImage *
        image) {
        MBProgressHUD *HUD = [[MBProgressHUD alloc] initWithView:self.view];
        [self.view addSubview:HUD];

        HUD.customView = [[UIImageView alloc]initWithImage:image];
        HUD.mode = MBProgressHUDModeCustomView;
        HUD.labelText = @"Vrata";
        [HUD show:YES];
        [HUD hide:YES afterDelay:3];

    } failureBlock:^(NSError *error) {
        NSLog(@"Error");
    }];
}
```

3.2 Strežnik

Pri načrtovanju sistema Vratar smo si za cilj zadali izdelati hiter, varen, zanesljiv in odziven sistem, ki je zmožen komunicirati s senzorji in ostalo strojno opremo v realnem času. Implementacijo dvosmerne komunikacije v realnem času smo dosegli z uporabo protokola WebSocket. Pri izbiri spletnega strežnika smo se odločili za strežnik Nginx, ki je izdan pod odprtokodno licenco. Glavna razloga za izbiro tega strežnika sta bila predvsem preprosta konfiguracija in učinkovitost. Strežnik sicer ne podpira protokola WebSocket, je pa sposoben zahtevke, ki uporabljajo ta protokol, posredovati drugemu strežniku. Konfigurirali smo ga torej tako, da se obnaša kot obratno-posredniški strežnik (angl. reverse proxy server). Zahtevke

posreduje ciljnemu spletnemu strežniku, ki smo ga implementirali z uporabo programske platforme Node.js.

Strežniške aplikacije smo namestili na računalnik Raspberry Pi, na katerem je nameščen operacijski sistem Raspbian [33]. Ta temelji na operacijskem sistemu Debian in je optimiziran za uporabo strojne opreme računalnika Raspberry Pi.

3.2.1 Obratno-posredniški strežnik Nginx

Strežnik Nginx je zmogljiv odprtokodni spletni in obratno-posredniški strežnik. Znan je po tem, da je stabilen, varen in enostaven za konfiguriranje. Zelo dobro se izkaže tudi pri obvladovanju večjega prometa.

V sistemu Vratar smo ga uporabili kot obratno-posredniški strežnik, ki skrbi za avtentikacijo odjemalca, šifriranje prometa in posredovanje zahtevkov ciljnemu spletnemu strežniku, ki je implementiran z uporabo programske platforme Node.js.

Pri postavitvi strežnika Nginx je bilo potrebno poskrbeti tudi za ustrezno varnost. Strežnik smo nastavili tako, da vsa komunikacija med odjemalcem in strežnikom poteka preko šifrirane povezave HTTPS (angl. Hypertext Transfer Protocol Secure). Dostop do strežnika smo zaščitili z uporabniškim imenom in geslom. V našem primeru se nam je ta zaščita zdela dovolj varna, saj je dostop do strežnika omejen na lokalno omrežje. Za to omejitev dostopa skrbi požarni zid, ki je nameščen na usmerjevalniku. Če želimo dostopati do sistema iz svetovnega spleta, se je potrebno najprej prijaviti v lokalno omrežje. Dostop do tega pa je mogoč le preko varne povezave – tunela Cisco IPSec VPN. Ta ima možnost uporabe funkcije vzpostavitve povezave na zahtevo (angl. VPN On-Demand), kar pomeni, da se bo povezava v lokalno omrežje po potrebi samodejno vzpostavila.

V primeru 3.5 je prikazan del konfiguracijske datoteke strežnika Nginx. Nastavitve strežnika so sledeče:

- strežnik posluša na vratih 80 (HTTP) in 443 (HTTPS)
  Zahtevki HTTP so preusmerjeni na šifrirano povezavo HTTPS.
- nastavitev datoteke s prijavnimi podatki
  Za prijavo potrebujemo uporabniško ime in geslo.
- posredniški strežnik za zahtevke WebSocket
  Zahtevki, ki uporabljajo protokol WebSocket, so preusmerjeni na drugi strežnik.

Primer 3.5: Konfiguracija strežnika Nginx.

```
server{
  listen 80;
  listen 443 ssl;
  server_name default;
  # omogoci osnovno avtentikacijo in nastavi lokacijo datoteke s prijavnimi
      podatki
  auth_basic "Restricted";
  auth_basic_user_file /var/www/default/.htpasswd;

  # pot do SSL certifikata, ki je potreben za SSL povezavo
  ssl_certificate /etc/nginx/ssl/server.crt;
  ssl_certificate_key /etc/nginx/ssl/server.key;

  # preusmeritev HTTP na HTTPS
  if ($ssl_protocol = "") {
    rewrite ^ https://$host$request_uri? permanent;
  }
...
  #preusmeritev zahtevkov na websocket streznik
  location /socket.io/ {
    proxy_pass http://localhost:8080;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection upgrade;
  }
...
}
```

3.2.2 Spletni strežnik

Za implementacijo spletnega strežnika smo uporabili programsko platformo Node.js [2], ki nam omogoča izvajanje programov na strani strežnika. Programi so napisani v programskem jeziku JavaScript. Platforma temelji na Googlovem pogonu V8 JavaScript. Namenjena je razvoju računsko nezahtevnih strežniških aplikacij, ki so hitre in učinkovite tudi pri velikem številu uporabnikov. K temu pripomore dogodkovno voden koncept programiranja, kar pomeni, da je program razdeljen na posamezne dele, ki se ob določenem dogodku izvršijo asinhrono.

Spletni strežnik, ki smo ga implementirali, uporablja za komunikacijo protokol WebSocket. Naloga spletnega strežnika je, da sprejema ukaze odjemalca (v našem primeru je to mobilna aplikacija mVratar) in jih posreduje vmesniku GPIO. Za komunikacijo z vmesnikom GPIO smo uporabili določene module.

V tem delu strežniške aplikacije je nastala razlika med žično in brezžično različico sistema Vratar. Pri žični različici je na vmesnik GPIO priključen vmesnik PiFace Digital. Za komunikacijo s tem vmesnikom smo uporabili modul `piface-node`. Pri brezžični različici pa je na vmesnik GPIO priključen krmilnik – ploščica Moteino. Tu komunikacija poteka preko serijske povezave z uporabo modula `SerialPort`.

Moduli

Zbirko funkcij, ki ponavadi predstavljajo sklop določene funkcionalnosti, lahko združimo v modul [11]. Moduli imajo ključen pomen pri razvoju kompleksnih aplikacij. Omogočajo nam, da v aplikacijo vključimo zunanje knjižnice, pripomorejo pa tudi pri sami organiziranosti programske kode. Programska platforma Node.js že v osnovi vsebuje pester nabor vgrajenih modulov, ki jih najdemo v mapi "lib". Dodatne module lahko poiščemo in namestimo s pomočjo upravljalnika paketov NPM (angl. Node Package Manager). Če želimo uporabiti določen modul, ga naložimo s klicem funkcije "require()".

Glavni moduli, ki smo jih uporabili v diplomski nalogi:

- socket.io
  Modul za postavitev spletnega strežnika, ki uporablja protokol WebSocket.

- piface-node
  Modul za komunikacijo z vmesnikom PiFace Digital.

- serialport
  Modul za delo s serijskim vmesnikom.

- apn
  Modul za povezovanje s storitvijo APNs (angl. Apple Push Notification service), ki omogoča pošiljanje potisnih obvestil.

- fs
  Modul za delo z datotečnim sistemom.

Modul socket.io

S pomočjo modula `socket.io` [23] smo realizirali spletni strežnik, ki za komunikacijo z odjemalcem uporablja protokol WebSocket. Protokol nam omogoča prenos podatkov v obe smeri hkrati (angl. full-duplex). Tako lahko poteka komunikacija med strežnikom in odjemalcem v realnem času. Primer 3.6 prikazuje uporabo modula v sistemu Vratar.

Primer 3.6: Primer uporabe modula socket.io v sistemu Vratar.

```
var socket = require('socket.io').listen(8080);
//avtorizacija - odobrimo samo zahteve, ki prihajajo iz Nginx streznika (
    lokalni IP)
socket.configure(function () {
  socket.set('authorization', function (handshake, accept) {
    if (handshake.address.address == "localhost" || handshake.address.
        address == "127.0.0.1")
      accept(null, true);
    else
      accept("Povezava zavrnjena!", false);
  });
});

socket.sockets.on('connection', function (socket) {
  var address = socket.handshake.address;
  console.log("Nova povezava: " + address.address + ":" + address.port);

  //stanje vrat (data = stevilka vrat)
  socket.on(kMsgDoorStatus, function (data) {
    socket.emit(kMsgLog, 'Preverjam stanje vrat...');
    serialPort.write(kMsgDoorStatus + kMsgDelimiter + data);
  });
...
});
```

Modul piface-node

Modul `piface-node` [22] smo uporabili pri žični različici sistema Vratar, in sicer za komunikacijo z vmesnikom PiFace Digital. Preko funkcij, ki so na voljo v modulu, smo brali stanje digitalnih vhodov, na katere sta priključeni magnetni stikali in krmilili rele za krmiljenje vrat. V primeru 3.7 vidimo preprost primer uporabe modula `piface-node` za branje in nastavljanje stanja priključkov vmesnika GPIO.

Primer 3.7: Primer branja in nastavljanja stanja priključka 0.

```
var pfio = require('piface-node');
pfio.init();
// prikljucku 0 nastavimo vrednost 1
pfio.digital_write(0,1);
// preberemo stanje prikljucka 0
var foo = pfio.digital_read(0);
pfio.deinit();
```

Modul serialport

V brezžični različici sistema Vratar smo vmesnik GPIO uporabili za serijsko povezavo s ploščico Moteino. Pri tem smo uporabili modul `serialport` [25]. Uporaba modula je prikazana v primeru 3.8.

Primer 3.8: Primer uporabe modula serialport.

```
//inicializacija serijskih vrat
var SerialPort = require("serialport").SerialPort;
var serialPort = new SerialPort("/dev/ttyAMA0", { baudrate : 9600, parser:
    serialport.parsers.readline("\n") });

//pocakamo, da se inicializira
serialPort.on("open", function () {
  console.log('Serijska povezava pripravljena');
  //Sinhronizacija casa
  console.log("Cas:" + "T" + Math.round(Date.now() / 1000));
  serialPort.write("T" + Math.round(Date.now() / 1000));

  serialPort.on("data", function(data) {
    console.log("Sprejeti podatki:" + data);
  });
});
```

Modul apn

Modul apn [21] omogoča pošiljanje potisnih obvestil preko storitve APNs. V sistemu Vratar smo ga uporabili za obveščanje odjemalca o določenih dogodkih (npr. predolgo odprta vrata). Funkcija, ki se pri tem uporablja, je prikazana v primeru 3.9.

Primer 3.9: Primer uporabe modula apn v sistemu Vratar.

```
//Apple Push Notification service
```

```
var apn = require('apn');
//nastavitve APN
const kApnCertFile = __dirname + '/cert.pem';
const kApnKeyFile = __dirname + '/key.pem';
const kApnGateway = 'gateway.sandbox.push.apple.com';
const kApnPort = 2195;
...
function sendNotification(message) {
  var options = {
      cert: kApnCertFile,
      key:  kApnKeyFile,
      gateway: kApnGateway,
      port: kApnPort
    }
  var apnConnection = new apn.Connection(options);

  var note = new apn.Notification();

  note.expiry = Math.floor(Date.now() / 1000) + 3600; // Expires 1 hour from
       now.
  //note.badge = 1;
  note.sound = "ping.aiff";
  note.alert = message;
  note.payload = {'messageFrom': 'Vratar'};

  apnConnection.pushNotification(note, tokens);
  console.log('posiljam:' + message);

  function log(type) {
    return function() {
      console.log(type, arguments);
    }
  }
  apnConnection.on('error', log('error'));
  apnConnection.on('transmitted', log('transmitted'));
  apnConnection.on('timeout', log('timeout'));
  apnConnection.on('connected', log('connected'));
  apnConnection.on('disconnected', log('disconnected'));
  apnConnection.on('socketError', log('socketError'));
  apnConnection.on('transmissionError', log('transmissionError'));
  apnConnection.on('cacheTooSmall', log('cacheTooSmall'));
}

function notifyDoorOpen(doorName, time) {
  sendNotification(doorName + "␣vrata␣so␣odprta␣vec␣kot␣" + time + "␣minut."
       );
}
```

Modul fs

Modul `fs` (File System) smo uporabili za delo z datotečnim sistemom. Seznam podatkov o napravah, ki so naročene na potisna obvestila, smo hranili v datoteki. Uporaba modula za branje in zapisovanje datoteke je prikazana v primeru 3.10.

Primer 3.10: Primer uporabe fs-modula.

```
//filesystem
var fs = require('fs');
//array zetonov naprav
var tokens = [];
const kTokensFile = 'tokens.txt';

//Nalozi seznam tokenov iz datoteke
function loadTokens() {
  if (fs.existsSync(kTokensFile)) {
    tokens = fs.readFileSync(kTokensFile).toString().split(",");
  }
}

//Shrani seznam tokenov v datoteko
function saveTokens() {
  //brisemo staro datoteko
  if (fs.existsSync(kTokensFile)) {
    fs.unlinkSync(kTokensFile);
  }

  for(t in tokens) {
    if (tokens[t] != '') {
      fs.appendFileSync(kTokensFile, tokens[t] + ",");
    }
  }
}
```

3.2.3 Bazni in podrejeni krmilnik (ploščica Moteino)

Razvoj programov oz. skic (angl. sketches) za razvojne ploščice Arduino poteka v razvojnem okolju Arduino [6]. Gre za odprtokodno razvojno okolje, ki je na voljo za operacijske sisteme Windows, Mac OS X in Linux, prenesemo pa ga lahko brezplačno iz proizvajalčeve spletne strani[1]. Skice lahko nalagamo neposredno na

[1] http://arduino.cc

krmilnik z uporabo USB-kabla. V diplomski nalogi smo uporabili ploščico Moteino, ki nima USB-priključka, zato je bilo potrebno pri nalaganju skic uporabiti vmesnik FTDI. Za razvoj skic smo uporabili različico razvojnega okolja Arduino 1.0.5.

V brezžični različici sistema Vratar smo uporabili bazni in podrejeni krmilnik, zato je bilo potrebno razviti dve skici ("VratarBazni.ino" in "VratarPodrejeni.ino"). Njuna naloga je procesiranje ukazov strežnika in posredovanje informacij o stanju sistema vrat strežniku. Komunikacija med njima poteka preko radijske povezave. Pri razvoju skic smo si pomagali z uporabo knjižnic. Sliki 3.8 in 3.9 prikazujeta diagram poteka za bazni in podrejeni krmilnik. Na sliki 3.10 pa je prikazan diagram poteka za pošiljanje sporočila s potrditvijo.

Knjižnice

Knjižnice nam nudijo dodatne funkcionalnosti za manipulacijo s podatki in strojno opremo. Podpora knjižnic, ki so napisane v programskem jeziku C in C++, je bila predstavljena v različici razvojnega okolja Arduino 004. Ob namestitvi okolja se namestijo tudi standardne knjižnice (tabela 3.1) [7], ki se uporabljajo pri najbolj pogostih funkcionalnostih (npr. krmiljenje servo motorjev, izpisovanje na prikazovalnik LCD itd.). Knjižnico lahko vključimo v skico preko menijske postavke "Skica→Uvozi knjižnico ...". Seveda pa lahko dodamo tudi svoje knjižnice.

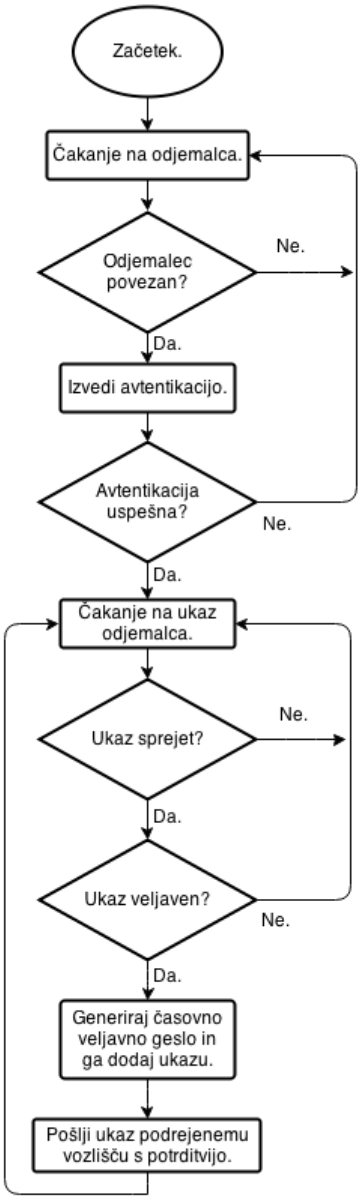
Pomembnejše knjižnice, ki smo jih uporabili v diplomski nalogi:

- RFM69
  Knjižnica za delo z RF oddajno-sprejemno enoto RFM69HW.

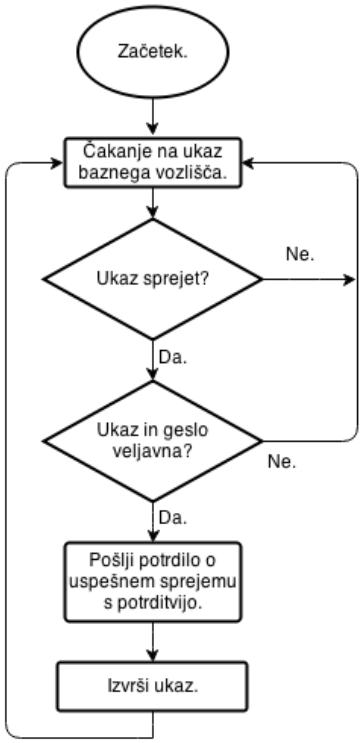
- LiquidCrystal
  Knjižnica za izpisovanje podatkov na prikazovalnik LCD.

- SPI
  Knjižnica za delo s serijskim vmesnikom.

- Time
  Knjižnica omogoča funkcionalnost za merjenje časa.



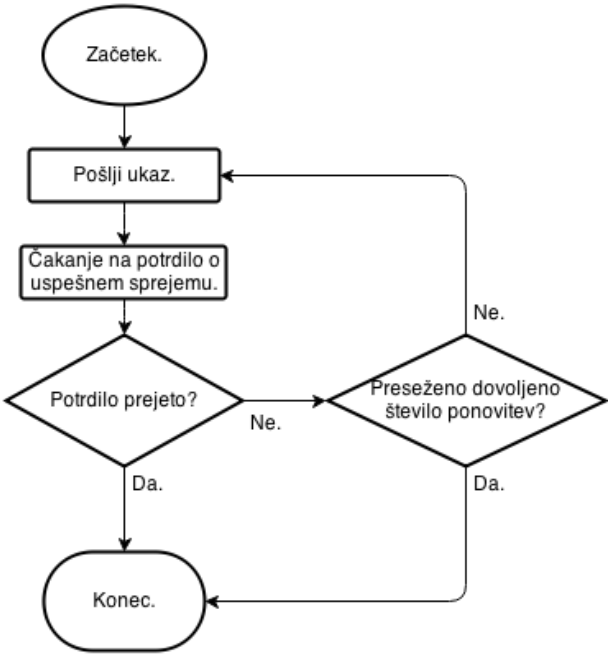
Slika 3.8: Diagram poteka – bazni krmilnik.



Slika 3.9: Diagram poteka – podrejeni krmilnik.

Knjižnica	Uporaba
EEPROM	Pisanje in branje na trajni pomnilnik.
Ethernet	Povezava z internetom z uporabo ščita Arduino Ethernet.
Firmata	Komunikacija s programsko opremo na gostiteljskem računalniku.
GSM	Povezava na GSM/GPRS-omrežje preko ščita GSM.
LiquidCrystal	Krmiljenje prikazovalnikov LCD.
SD	Pisanje in branje na SD-kartice.
Servo	Krmiljenje servo motorjev.
SPI	Komunikacija z napravami preko serijskega vmesnika.
SoftwareSerial	Serijska komunikacija preko poljubnih priključkov.
Stepper	Krmiljenje koračnih motorjev.
TFT	Risanje besedila, slik in ostalih likov na prikazovalnik TFT.
WiFi	Povezava z internetom z uporabo ščita Arduino WiFi.
Wire	Komunikacija z I^2C/TWI-napravami.

Tabela 3.1: Standardne knjižnice razvojnega okolja Arduino.



Slika 3.10: Diagram poteka - pošiljanje sporočila s potrditvijo.

Knjižnica RFM69

Knjižnico smo uporabili v brezžični različici sistema in sicer za komunikacijo z RF oddajno-sprejemno enoto RFM69HW [24]. Z uporabo osnovnih funkcij lahko vzpostavimo komunikacijo med posameznimi krmilniki (primer 3.11). Možna je komunikacija z največ 255 krmilniki na 256 različnih omrežjih. Največja dovoljena velikost sporočila je 61 bajtov (omejitev zaradi podpore strojne enkripcije AES). Porabo energije lahko zmanjšamo z uporabo funkcije “sleep”, ki postavi oddajno-sprejemno enoto v način nizke porabe. Poleg tega je možno prilagoditi moč oddajanja v 32 stopnjah. Funkcija za pošiljanje sporočil ima podporo za povratnice, s pomočjo katerih dobimo informacijo o uspešni oz. neuspešni dostavi sporočila. Nad prejetim sporočilom se izvede metoda za odkrivanje napak – CRC (angl. cyclic redundancy check). V primeru napak ali neuspešne dostave se sporočilo samodejno ponovno pošlje. Število ponovitev pošiljanja nastavimo s parametrom funkcije.

Primer 3.11: Pošiljanje sporočila z uporabo knjižnice RFM69.

```
#include <RFM69.h>
#include <SPI.h>
RFM69 radio;

void setup() {
  delay(10);
  // frekvenca 433Mhz, krmilnik 1, omrezje 100
  radio.initialize(RF69_433MHZ,1,100);
  // samo za RFM69HW
  radio.setHighPower();
  radio.encrypt("Kljuc");
}

void loop() {
  // posljemo s ponovitvijo sporocilo "test", dolzine 4, v krmilnik 2
  if (radio.sendWithRetry(2, "test", 4))
    Serial.print("Sporocilo poslano.");
  else
    Serial.print("Napaka.");
  delay(5000);
}
```

Knjižnica LiquidCrystal

Pri izpisovanju sporočil na prikazovalnik LCD smo si pomagali s knjižnico "LiquidCrystal". Knjižnica omogoča krmiljenje prikazovalnikov, ki so združljivi z gonilnikom za krmilnik Hitachi HD44780. Ponavadi jih prepoznamo po vmesniku s šestnajstimi priključki [8, 9].

Primer 3.12: Izpisovanje besedila na prikazovalnik LCD z uporabo knjižnice LiquidCrystal.

```
#include <LiquidCrystal.h>

// inicializacija s prikljucki, na katere je prikljucen LCD
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup() {
  // dolocimo stevilo kolon in vrstic prikazovalnika
  lcd.begin(16, 2);
  // izpis sporocila na LCD
  lcd.print("hello, world!");
}
```

```
void loop() {
  // kurzor postavimo na zacetek druge vrstice
  lcd.setCursor(0, 1);
  // izpis sporocila na LCD
  lcd.print("Arduino");
}
```

Knjižnica SPI

Knjižnico SPI smo uporabili na baznem krmilniku, in sicer za komunikacijo s spletnim strežnikom, ki je potekala preko serijskega vmesnika. Primer 3.13 prikazuje uporabo knjižnice za branje iz serijskega vmesnika.

Primer 3.13: Branje iz serijskega vmesnika.

```
#include <SPI.h>
char data[32];
byte dataLength = 0;
...
  Serial.begin(9600);
  //znak za novo vrstico "10", najvecja dolzina 32
  dataLength = Serial.readBytesUntil(10, data, 32);
```

Knjižnica Time

Časovne funkcije smo potrebovali pri implementaciji algoritma TOTP (angl. Time-based One-time Password Algorithm), ki pri izračunu žetona potrebuje podatek o trenutnem času. Ker razvojna ploščica Arduino nima svoje ure realnega časa, je potrebno uporabiti strojno ali programsko rešitev. Knjižnica "Time" [32] nam za vodenje ure realnega časa ponuja programsko rešitev. Prednost te rešitve je, da nam ni potrebno kupovati dodatne strojne opreme z uro. Slaba stran pa je ta, da je potrebno poskrbeti za sinhronizacijo ure, ki ponavadi poteka preko serijskega vmesnika. Tu je strojna rešitev v prednosti, saj sinhronizacija ni potrebna, ura pa je usklajena tudi po ponovnem zagonu razvojne ploščice Arduino. V primeru 3.14 je prikazana funkcija, ki se uporablja pri sinhronizaciji časa preko serijskega vmesnika. Sporočilo za sinhronizacijo je sestavljeno iz črke "T" in desetih cifer, ki predstavljajo čas, merjen v sekundah od 1. 1. 1970.

Primer 3.14: Sinhronizacija časa preko serijskega vmesnika.

```
#include <Time.h>
```

```
// dolzina sporocila za sinhronizacijo (glava + Unix timestamp)
#define TIME_MSG_LEN  11
// glava sporocila
#define TIME_HEADER  'T'
...
void processSyncMessage() {
  // preverimo, ce na serijski povezavi caka sporocilo za sinhronizacijo
  while(Serial.available() >=  TIME_MSG_LEN ){
    char c = Serial.read();
    if( c == TIME_HEADER ) {
      time_t pctime = 0;
      for(int i=0; i < TIME_MSG_LEN -1; i++){
        c = Serial.read();
        if( c >= '0' && c <= '9'){
          // konverzija v stevilo
          pctime = (10 * pctime) + (c - '0');
        }
      }
      // na krmilniku nastavimo uro, ki smo jo prebrali iz sporocila za
          sinhronizacijo
      setTime(pctime);
    }
  }
}
```

Generiranje časovno veljavnega gesla

Običajno daljinski upravljalniki garažnih vrat za zaščito ukazov uporabljajo sistem vrtečih kod, ki nudi visoko zaščito pred krajo kode. Ta se namreč po vsakem ukazu menja. S tem so onemogočeni napadi s prisluškovanjem, kjer napadalec posname signal in si s kasnejšim predvajanjem signala odpre vrata.

Na podoben način smo zaščitili komunikacijo med krmilniki sistema Vratar. Glavna razlika je v tem, da se tu uporabljajo gesla, ki imajo časovno omejeno veljavnost. Za generiranje gesel smo uporabili algoritem TOTP (angl. Time-based One-time Password Algorithm). Algoritem nam na podlagi trenutnega časa in skrivnega ključa z uporabo kriptografske funkcije zgenerira geslo, ki je veljavno v določenem časovnem okvirju (v našem primeru je to 30 sekund). Pogoj za pravilno delovanje je, da imata krmilnika, ki med sabo komunicirata, isti skrivni ključ ter sinhroniziran čas. Sinhronizacija časa se izvrši v določenih časovnih intervalih oz. takoj po priključitvi podrejenega krmilnika v sistem.

Poglavje 4

Sklepne ugotovitve

V diplomski nalogi smo opisali razvoj sistema Vratar, katerega namen je krmiljenje garažnih vrat s pomočjo mobilne aplikacije. Sistem Vratar temelji na principu strežnik – odjemalec in je sestavljen iz strojnega ter programskega dela, zato smo tudi diplomsko nalogo razdelili na dva dela.

V prvem delu je predstavljena strojna oprema, ki sestavlja strežniški del sistema Vratar. Temelj sistema Vratar je računalnik Raspberry Pi, na katerega je nameščena programska oprema strežnika, nanj pa so vezane tudi ostale strojne komponente sistema.

Uspeli smo realizirati dve različici sistema Vratar. V prvotni različici smo uporabili vmesnik PiFace Digital, ki nam je omogočil krmiljenje največ dvojih vrat. Zaradi žičnih povezav med računalnikom Raspberry Pi in strojno opremo se je ta različica izkazala za primerno le v primeru, če je celotna strojna oprema strežnika nameščena v neposredni bližini vrat. Ker pa to običajno ni izvedljivo, smo se odločili še za razvoj brezžične različice, ki za komunikacijo med posameznimi krmilniki uporablja radijsko povezavo. To verzijo smo realizirali z uporabo izpeljanke razvojne ploščice Arduino, ki je opremljena z RF-oddajno-sprejemno enoto. Sheme tiskanega vezja razvojne ploščice so javno dostopne, zato smo se odločili, da si jo izdelamo sami. V tem delu smo se soočili s spajkanjem komponent za površinsko montažo, kar se je izkazalo za dokaj zahtevno opravilo.

V obdobju testiranja brezžične različice sistema nismo zaznali nobenih težav. Delovanje je bilo zelo zanesljivo in učinkovito. Dobra lastnost tega sistema je, da

je modularen in zato enostavno razširljiv. Sistem Vratar bi lahko z uporabo dodatnih senzorjev (npr. senzor za zaznavanje gibanja, temperature, dima, vlažnosti itd.) z lahkoto nadgradili v sistem pametne hiše.

Drugi del diplomske naloge je namenjen predstavitvi programskega dela sistema Vratar. Tu je predstavljen tudi razvoj odjemalca – mobilne aplikacije za operacijski sistem iOS, ki smo jo poimenovali mVratar. Z intuitivnim vmesnikom za krmiljenje vrat smo uspeli zagotoviti dobro uporabniško izkušnjo. Implementirali smo tudi opcijo samodejnega odpiranja vrat na podlagi lokacije odjemalca, ki nam omogoča odpiranje vrat med vožnjo, brez poseganja po mobilni napravi. V primeru, da se vrata niso zaprla ali pa smo jih enostavno pozabili zapreti, nas sistem po določenem času o tem obvesti preko potisnega obvestila. Stanje vrat je možno preveriti tudi vizualno, z uporabo funkcionalnosti prikaza slike, ki jo sistem zajame s pomočjo kamere.

Aplikacija mVratar se je po mesecu dni uporabe v praksi izkazala za zelo uporabno, saj smo z njeno uporabo brez težav nadomestili daljinski upravljalnik. V morebitnem nadaljnjem razvoju bi lahko nadgradili zaščito prijave na strežnik z uporabo certifikatov. V primeru nadgradnje sistema v sistem pametne hiše pa bi lahko aplikacijo opremili z dodatnimi kontrolami in informacijami, ki bi jih pridobili s pomočjo raznih senzorjev.

Literatura

[1] Massimo Banzi, Getting Started with Arduino (Make: Projects), str. 17–18, 2012.

[2] Don Nguyen, Jump Start Node.js, str. 2–4, 2012.

[3] Maik Schmidt, Raspberry Pi: A Quick-Start Guide, str.13, 2012.

[4] Standard LCD 20x4 + extras - white on blue. Dostopno na: http://www.adafruit.com/products/198 (Povzeto 04.03.2014).

[5] APC Back-UPS 500, 230V. Dostopno na: http://www.apc.com/resource/include/techspec_index.cfm?base_sku=bk500ei (Povzeto 04.03.2014).

[6] Arduino Development Environment. Dostopno na: http://arduino.cc/en/Guide/Environment (Povzeto 09.03.2014).

[7] Arduino - Libraries. Dostopno na: http://arduino.cc/en/Reference/Libraries (Povzeto 11.03.2014).

[8] LiquidCrystal Library. Dostopno na: http://arduino.cc/en/Reference/LiquidCrystal (Povzeto 12.03.2014).

[9] LiquidCrystal - "Hello World!". Dostopno na: http://arduino.cc/en/Tutorial/LiquidCrystal (Povzeto 12.03.2014).

[10] Slika: Mrežna kamera Axis 213 PTZ. Dostopno na: http://www.axis.com/files/image_gallery/213/213_CEILING_LEFT_UP.JPG (03.05.2014).

[11] Node: Modules and npm - Mixu's Node book. Dostopno na: http://book.mixu.net/node/ch8.html (Povzeto 21.03.2014).

[12] Ultra S Garage Door Electric Operator. Dostopno na: http://www.crawforddanmark.dk/files/manager/documents/ultra_s_1.pdf (Povzeto 16.04.2014).

[13] Slika: Pogon garažnih vrat - Crawford Ultra S. Dostopno na: http://dev.terrasoase.nl/images/stories/UltraS1.jpg (03.05.2014).

[14] Hall Effect Sensor and How Magnets Make It Works. Dostopno na: http://www.electronics-tutorials.ws/electromagnetism/hall-effect.html (Povzeto 26.02.2014).

[15] Raspberry Pi Single Board Computer. Dostopno na: http://www.element14.com/community/docs/DOC-42993/l/raspberry-pi-single-board-computer (Povzeto 28.01.2014).

[16] Slika: Računalnik Raspberry Pi Model-B. Dostopno na: http://en.wikipedia.org/wiki/File:RaspberryPi.jpg (03.05.2014).

[17] Hall effect. Dostopno na: http://en.wikipedia.org/wiki/Hall_effect (Povzeto 26.02.2014).

[18] Hitachi HD44780 LCD controller. Dostopno na: http://en.wikipedia.org/wiki/Hitachi_HD44780_LCD_controller (Povzeto 04.03.2014).

[19] I^2C. Dostopno na: http://en.wikipedia.org/wiki/I^2C (Povzeto 24.02.2014).

[20] Slika: Ploščica Moteino. Dostopno na: http://farm8.staticflickr.com/7415/10702055256_498be59eba_o.jpg (03.05.2014).

[21] node-apn. Dostopno na: https://github.com/argon/node-apn (Povzeto 02.04.2014).

[22] piface-node. Dostopno na: https://github.com/darrylhodgins/piface-node (Povzeto 02.04.2014).

[23] socket.io. Dostopno na:
https://github.com/LearnBoost/socket.io (Povzeto 02.04.2014).

[24] RFM69 Library. Dostopno na:
https://github.com/LowPowerLab/RFM69 (Povzeto 12.03.2014).

[25] node-serialport. Dostopno na:
https://github.com/voodootikigod/node-serialport (Povzeto 02.04.2014).

[26] What is Arduino? Learn About This Open-Source Electronics Platform. Dostopno na:
http://www.howtogeek.com/65963/what-is-arduino/ (Povzeto 01.03.2014).

[27] Slika: Magnetno stikalo. Dostopno na:
http://inspiredled.com/image/cache/data/products/accessories/magswitch(2)-500x350.jpg (03.05.2014).

[28] Adafruit's Raspberry Pi Lesson 4. GPIO Setup. Dostopno na:
http://learn.adafruit.com/adafruits-raspberry-pi-lesson-4-gpio-setup/overview (Povzeto 24.02.2014).

[29] All about Moteino. Dostopno na:
http://lowpowerlab.com/moteino/ (Povzeto 02.03.2014).

[30] Slika: Vmesnik PiFace Digital. Dostopno na:
https://www.modmypi.com/image/cache/data/raspberry-pi-expansion-boards/piface-digital-800x800.jpg (03.05.2014).

[31] PiFace Digital I/O Expansion Board. Dostopno na:
https://www.modmypi.com/piface-digital-raspberry-pi-expansion-board (Povzeto 24.02.2014).

[32] Arduino Time library. Dostopno na:
http://playground.arduino.cc/Code/Time (Povzeto 12.03.2014).

[33] Raspbian. Dostopno na:
http://www.raspbian.org (Povzeto 03.05.2014).

[34] SPI. Dostopno na:
http://sl.wikipedia.org/wiki/SPI (Povzeto 24.02.2014).

[35] System on a Chip: what you need to know about SoCs. Dostopno na: http://www.techradar.com/news/computing/pc/system-on-a-chip-what-you-need-to-know-about-socs-1147235 (Povzeto 28.01.2014).